Guaranteed Avoidance of Unpredictable, Dynamically Constrained Obstacles using Velocity Obstacle Sets

by

Albert Wu

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Master of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2011

Author .....
Department of Aeronautics and Astronautics
May 19, 2011

Certified by .....
Jonathan P. How
Richard C. Maclaurin Professor of Aeronautics and Astronautics
Thesis Supervisor

Accepted by .....
Professor Eytan H. Modiano
Associate Professor of Aeronautics and Astronautics
Chair, Graduate Program Committee

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
JUN 2011		**00-00-2011 to 00-00-2011**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Guaranteed Avoidance of Unpredictable, Dynamically Constrained Obstacles using Velocity Obstacle Sets	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Massachusetts Institute of Technology,Cambridge,MA,02139**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Dynamic obstacle avoidance is an important, ubiquitous, and often challenging prob- lem for autonomous mobile robots. This thesis presents a new method to guarantee collision avoidance with respect to moving obstacles that have constrained dynamics but move unpredictably. Velocity Obstacles have been widely used to plan trajectories that avoid collisions with obstacles under the assumption that the path of the objects are either known or can be accurately predicted ahead of time. However, for real systems, this predicted path will typically only be accurate over short time-horizons. To achieve safety over longer time periods, the method introduced here instead con- siders the set of all reachable points by an obstacle assuming that the dynamics t the unicycle model, which has known constant forward speed and a maximum turn rate (sometimes called the Dubins car model). This thesis extends the Velocity Obstacle formulation by using reachability sets in place of a single known" trajectory to nd matching constraints in velocity space called Velocity Obstacle Sets. The Velocity Obstacle Set for each obstacle is equivalent to the union of all velocity obstacles corresponding to any dynamically feasible future trajectory, given the obstacle's current state. This region remains bounded as the time horizon is increased to in nity, and by choosing control inputs that lie outside of these Velocity Obstacle Sets, it is guaranteed that the host agent can always actively avoid collisions with the obstacles, even without knowing their exact future paths. It thus follows that, subject to certain initial conditions, an iterative planner under these constraints guarantees safety for all time. Finally, the an iterative planner is repeatedly tested and analyzed in simulation under various conditions. If the time horizon is set to some nite value, the guaranteed collision avoidance is lost, but the planned trajectories generally become more direct. This e ect of varying this time scale also depends on the presence of static obstacles in the environment and on the dynamic limitations of the host robot.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**116**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

Guaranteed Avoidance of Unpredictable, Dynamically Constrained Obstacles using Velocity Obstacle Sets

by

Albert Wu

Submitted to the Department of Aeronautics and Astronautics
on May 19, 2011, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

Abstract

Dynamic obstacle avoidance is an important, ubiquitous, and often challenging problem for autonomous mobile robots. This thesis presents a new method to guarantee collision avoidance with respect to moving obstacles that have constrained dynamics but move unpredictably. Velocity Obstacles have been widely used to plan trajectories that avoid collisions with obstacles under the assumption that the path of the objects are either known or can be accurately predicted ahead of time. However, for real systems, this predicted path will typically only be accurate over short time-horizons. To achieve safety over longer time periods, the method introduced here instead considers the set of all reachable points by an obstacle assuming that the dynamics fit the unicycle model, which has known constant forward speed and a maximum turn rate (sometimes called the Dubins car model).

This thesis extends the Velocity Obstacle formulation by using reachability sets in place of a single "known" trajectory to find matching constraints in velocity space, called *Velocity Obstacle Sets*. The Velocity Obstacle Set for each obstacle is equivalent to the union of all velocity obstacles corresponding to any dynamically feasible future trajectory, given the obstacle's current state. This region remains bounded as the time horizon is increased to infinity, and by choosing control inputs that lie outside of these Velocity Obstacle Sets, it is guaranteed that the host agent can always actively avoid collisions with the obstacles, even without knowing their exact future paths. It thus follows that, subject to certain initial conditions, an iterative planner under these constraints guarantees safety for all time.

Finally, the an iterative planner is repeatedly tested and analyzed in simulation under various conditions. If the time horizon is set to some finite value, the guaranteed collision avoidance is lost, but the planned trajectories generally become more direct. This effect of varying this time scale also depends on the presence of static obstacles in the environment and on the dynamic limitations of the host robot.

Thesis Supervisor: Jonathan P. How
Title: Richard C. Maclaurin Professor of Aeronautics and Astronautics

Acknowledgments

First, I owe deep thanks to my advisor, Professor Jonathan How, for his guidance over the last two years. His feedback and insights have been greatly beneficial to my research and to my academic career, and his consistent example of hard work and professionalism has demonstrated what it takes to achieve excellence. I also sincerely thank Dr. Luca Bertucelli for the attentive mentorship he has provided me.

I would like to thank the members of the ACL for their assistance and for their friendship. Much of what I have learned these two years had come directly from my fellow lab-members who have never been too busy to help me out, despite the heavy burdens of their own responsibilities. I would especially like to acknowledge Luke Johnson, Andrew Whitten, Dan Levine, Buddy Michini, and Sameera Ponda for their camaraderie that has made my time here enjoyable and memorable.

I thank the members of MIT Syncopasian who have provided warm respite from the academic rigors of my program, and I thank my friends who have kept in touch over the distance, doing much to keep me positive. I would like to specifically thank Dan Kahn and Derek Goto for their input and expertise that contributed directly to this thesis. I owe more than I can say to my parents, who have always been there to guide and aid me, and who have taught me so much through example. A very special thanks goes to my fiancee Beverly for her tireless support and invaluable companionship; I could not have done any of this without her.

This work was funded by AFOSR # FA9550-08-1-0086.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Dynamic obstacle avoidance is an important, ubiquitous, and often challenging problem for autonomous mobile robots. The characteristics of specific scenarios call for different sets of assumptions and different collision avoidance algorithms.

In situations with long time-scales and significant uncertainty about the future, accurately generating an entire plan for the host robot at once may not be feasible. Instead, on-line motion planners can be used to create or modify the trajectory as new information becomes available. The complexity of the problem often prohibits these approaches from being able to guarantee finding a path to the goal. Instead, these planners iteratively extend the trajectory along the most promising segments, while ideally maintaining safety with respect to the obstacles in the environment.

Safety guarantees for such planners are most commonly achieved by defining safe states and restricting the planner to work only with trajectories composed of such states. Safe states are those that do not violate the imposed collision constraints and can transition into another safe state. For example, in structured environments in which the other vehicles practice reasonable collision avoidance, coming to a complete stop and staying stationary can be considered reaching an invariant safe state (which can be propagated indefinitely), so any state that can make this transition would be deemed safe [1, 2]. However, if the dynamic obstacles exhibit *unpredictable behavior*,

as often is the case in many real-world environments, it becomes much more difficult to define sufficient conditions that allow safe states to be propagated forward in time. The host vehicle may be struck by other vehicles if it stays at rest, or it may become surrounded by multiple other vehicles such that collision becomes inevitable.

Other popular collision avoidance approaches include being able to accurately anticipate imminent collisions such that, with an understanding of the dynamical capabilities of the host robot, timely evasive actions may be taken. However, with multiple unpredictable robots, forecasting situations like becoming from which collisions are not immediate but eventually inevitable (like becoming surrounded) is not straight-forward, and formulations like the collision-cone approach [3] cannot guarantee proper handling of these problematic cases.

This thesis presents a method for finding velocity-space constraints for an on-line planner that **guarantees infinite horizon safety in an environment with multiple obstacles that have constrained dynamics but can move unpredictably,** a scenario that has so far been difficult for techniques in the existing literature. This safety guarantee is achieved by combining the reachability set as a function of time for objects subject to these dynamics [4] with the velocity obstacle concept [5].

1.2 Problem Statement

The goal is to safely navigate a single host robot through a 2D environment with multiple dynamic obstacles. Each obstacle moves with unicycle dynamics (sometimes referred to as Dubins dynamics); it has a fixed forward speed v, and it can turn at up to some maximum rate ω:

$$\dot{\vec{x}} = v[\cos(\theta) \quad \sin(\theta)]^T,$$
$$|\dot{\theta}| \leq \omega,$$
$$\dot{v} = 0.$$

Equivalently, the obstacle can be described as having a minimum turning radius $\rho = v/\omega$. The host robot may be subject to arbitrary dynamic constraints. All the vehicles are discs, and a collision occurs between if the distance between two vehicles is less than collision radius r. Given the values of these parameters and the ability to measure exactly the current location and orientation of all dynamic obstacles, the host robot must travel in such a way that it will never come in contact with an obstacle, without any additional information. The host robot is given a series of way-points that it attempts to reach, if it can do so without jeopardizing its safety.

1.3 Previous Work

An expansive collection of path planning algorithms has been built as varied techniques geared towards a gamut of problem statements have been designed, modified, and combined. [6–11] This section will briefly cover the existing approaches that are most closely related to the planning problem posed in this thesis. To the best of our knowledge, there is no solution in the current literature that gives infinite horizon safety in the presence of unpredictable obstacles.

1.3.1 Velocity Obstacle Based Formulations

Using the original formulation of the velocity obstacle [5], one can find single velocity trajectories that are guaranteed collision-free, given the exact trajectory of the obstacles for some time-scale. This time-scale could be infinite, but that would unrealistically require that all obstacle trajectories be known perfectly for all time. For some applications in predictable environments [12–14], this is an effective means of collision avoidance, but it is unsuitable for guaranteeing long term safety with respect to unpredictable obstacles. Nonetheless, this work introduces the idea of using velocity-space constraints to concisely represent collision avoidance conditions, which plays a central role in the algorithm derived in this thesis and is reviewed in detail in Section 2.2.

Various extensions and modifications to the original velocity-obstacle approach

have been made. Instead of assuming a known infinite trajectory, in [15], the motion of the obstacles are predicted for a finite duration with an associated uncertainty. Velocity obstacles are computed after growing the obstacles by the uncertainty, and then used to plan safe finite segments. Since the time-scale is finite, there is no guarantee that a trajectory can be continued safely when a re-plan is necessary. Further, according to the authors of Ref. [15], growing obstacles by the uncertainty often results in all reachable velocities creating potential collisions. Various metrics can be used to select a velocity with relatively low likelihood of collision, but this would still be unacceptable if safety is to be guaranteed.

In [16], the authors appeal to a safe state argument. They use the dynamics of the host robot and of each obstacle to calculate an optimal time horizon, such that *inevitable collision states* [17] are avoided by heeding the bounds of the corresponding finite-time velocity obstacle. Firstly, this method still requires the trajectories of the dynamic obstacles be known at least up to some finite horizon, but this information is not always available in the most general settings. Further, it is argued that "any velocity that does not penetrate this [finite, optimal time horizon] velocity obstacle should allow sufficient time under the given control authority to avoid collision." But this in fact fails to account for the interaction of the constraints imposed by multiple obstacles. The extremal trajectories that are computed to dodge collisions with an obstacle may interfere with neighboring obstacles, so a velocity that respects the optimal horizon velocity obstacle of one object may in fact be forced to collide with a different obstacle. Thus, this method does not guarantee infinite horizon safety.

1.3.2 Various other formulations

In [3], "collision cones" are defined to predict collisions between objects of arbitrary size and shape, if the objects maintain their current velocities. For rigid bodies, it is then possible to compute the necessary adjustments to the velocity of the host robot such that anticipated collision is averted. Based on the dynamic capabilities of the robot, avoidance maneuvers can be computed such that it is always possible to remain the collision cone. However, much like the issue with [16] discussed earlier,

the feasibility of these maneuvers is guaranteed only for single obstacle case; the possibility of multiple obstacles imposing constraints that cannot be simultaneously satisfied is not explicitly accounted for. This problem arising from multiple obstacles cannot be solved by defining a single entity composed of the collective configurations of all the obstacles, since their independent motions would violate the rigid body assumption.

Other work includes [18], in which safe, analytic trajectories are found given robot and obstacle dynamic constraints, but only on a finite horizon with a given endpoints and without conditions for propagating safety. Thus, this algorithm cannot practically handle large planning problems with long time horizons.

In [19–22], guaranteed safety is achieved for multi-agent systems in which all the agents use the same collision avoidance policy with feasible invariant sets. These frameworks rely on having coordinated authority over all agents in the environment, and therefore cannot handle external unpredictable agents.

1.4 Summary of Contributions

This thesis develops a framework for guaranteeing infinite horizon collision avoidance that does not rely on any prediction of the obstacles' motion. This is done by mapping the reachable sets of the dynamic obstacles into velocity space to form Velocity Obstacle Sets which are then used to define safe, invariant single-velocity trajectories.

The safety guarantee is analytically derived and proved, and the resulting iterative planner is thoroughly demonstrated in various forms under different simulated conditions. It is also shown that alternative finite horizon velocity obstacle set formulations may be implemented as reactive planners that lack long-term safety guarantees but may be better suited for certain tasks, depending on the characteristics of the host vehicle.

1.5 Thesis Overview

The remainder of this thesis is organized as follows: Chapter 2 reviews and builds upon key concepts from existing literature that are critical to the new velocity obstacle set algorithm. These topics included reachability sets, velocity obstacles, and invariant safe states. Chapter 3 defines and derives velocity obstacle sets. A method is presented for efficiently computing a concise representation of constraints as a function of the current obstacle states. Safety guarantees on the infinite horizon are then discussed in Chapter 4. Chapter 5 describes how the velocity obstacle sets can be used in iterative planners. The behaviors of a basic infinite-horizon planner are demonstrated in simulation, and the use of infinite horizon planners and finite horizon planners of various lengths are then thoroughly compared for a variety of scenarios.

Chapter 2

Background

2.1 Reachability and Collision Regions

This section will find the boundaries of regions in physical space that may result in a collision with a given obstacle, as a function of time. For example, if the location of a circular obstacle is known for some future time, the *collision region* at that time is a circle centered on the known obstacle location, with radius equal to the sum of the obstacle and host vehicle radii. If the center of the host vehicle is within this region at that time, there will be contact with the given obstacle. These regions of physical space to avoid will be converted into velocity space constraints in Chapter 3.

2.1.1 Reachability

If it is not possible to know exactly the location of the obstacle as a function of time, the *reachable set* can be used instead to generate the collision region. This is the union of all possible locations of the obstacle at some given time. By avoiding the entire reachable set of the obstacle, the host robot avoids any possible contact.

In [4], the authors find "the set of all possible positions" for a particle that "moves in the plane with constant speed and subject to an upper bound on the curvature of its path." These dynamics match the standard unicycle model used here, and the results can be directly applied to find the reachable set of a given obstacle as a

Table 2.1: Summary of important terms and parameters used.

symbol	Section	Meaning
VO	2.2	Velocity obstacle: constraint in velocity space given trajectory of obstacle.
$VOS(t_0, t_f)$	3.1	Velocity obstacle set: constraint in velocity space given reachability of obstacle. Defined over a time window as $\bigcup \text{SVR}(t)\ \forall t \in [t_0, t_f]$.
$VOS(t_0, \infty)$	3.3.2	Infinite horizon VOS. Defined for all future times.
τ	4	Time horizon of VOS, $\tau = t_f - t_0$.
v	1.2	Constant forward speed of moving obstacles.
ω	1.2	Maximum turn rate of moving obstacles.
ρ	1.2	Minimum turning radius of obstacle, $\rho = \frac{v}{\omega}$.
r	1.2	Collision radius. Sum of host vehicle radius and obstacle radius.
$\text{CR}(t)$	2.1.2	Collision region: reachable set grown by collision radius, i.e., set of all points within r of reachable set.
$\text{SCR}(t)$	2.1.2	Simplified collision region: expanded version of CR(t) for simplicity; contains all points that could contact obstacle.
$S_{1,2,3,4,5}(\theta, t)$	2.1.2	Arcs that parametrically define respective segments of boundary of $\text{SCR}(t)$.
$\text{SVR}(t)$	3.1	Simplified velocity region: velocity space constraint associated with $\text{SCR}(t)$; $\text{SVR}(t) = \frac{1}{t}\text{SCR}(t)$.
$C_(t)$	3.1	Boundary of $\text{SVR}(t)$.
$Q_{1,2,3,4,5}(\theta, t)$	3.1	Arcs that parametrically define respective segments of $C(t)$.
$\hat{n}\|_P(\theta)$	3.1	Normal vector to $C(t)$ at point P.
$B_{\text{VOS}}(t_0, t_f)$	3.1	Boundary of $VOS(t_0, t_f)$
δt	5.2.1	Discrete time-step at which dynamics are propagated in simulation.
ΔT	5.2.1	Interval at which re-planning occurs.
$\Delta\theta$	5.2.2	Maximum instantaneous change of robot heading allowed at each re-plan.
ω_{max}	5.2.2	Equivalent turn-rate constraint of host vehicle represented by imposing limits on $\Delta\theta$

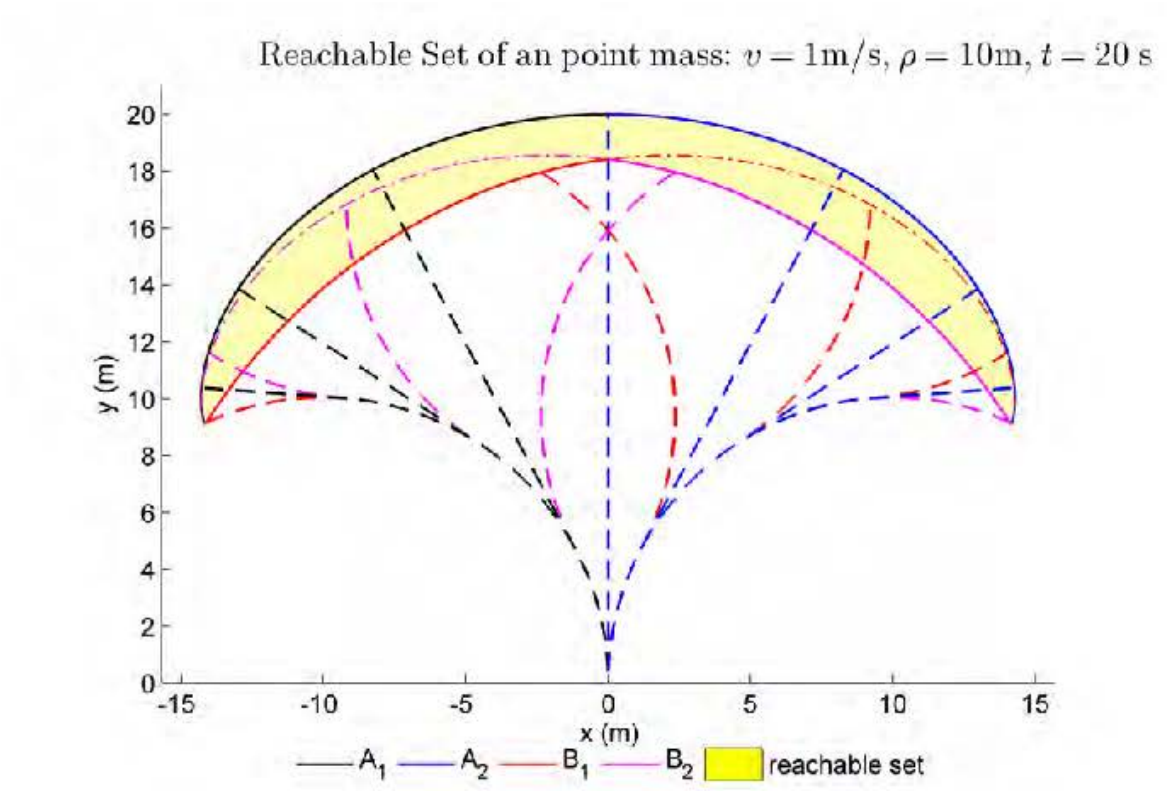

Figure 2-1: Example reachable set for $v = 1m/s, \rho = 10m, t = 20s$. The reachable region is bound by segments of the 4 curves A_1, A_2, B_1, B_2. Corresponding trajectories to points on these curves are shown in dashed lines.

function of time. Without loss of generality, consider an obstacle P that starts at the origin with *heading* (clockwise angle measured from the y axis) $\theta = 0$ at time $t = 0$. P moves with constant speed v and maximal turn rate ω:

$$\dot{\vec{x}}|_P = v[\sin(\theta) \quad \cos(\theta)]^T,$$
$$|\dot{\theta}| \leq \omega,$$
$$\dot{v} = 0.$$

Equivalently, P has a minimal turning radius $\rho = v/\omega$.

It is shown in [4] that the reachable region at time t is bounded by 4 parametric

curves (Figure 2-1). $A_{1,2}$ are given by

$$A_1(\theta, t) = \begin{bmatrix} -\rho(1 - \cos\theta) + (vt + \rho\theta)\sin\theta \\ -\rho\sin\theta + (vt + \rho\theta)\cos\theta \end{bmatrix} \tag{2.1}$$

$$A_2(\theta, t) = \begin{bmatrix} \rho(1 - \cos\theta) + (vt - \rho\theta)\sin\theta \\ \rho\sin\theta + (vt - \rho\theta)\cos\theta \end{bmatrix}, \tag{2.2}$$

with $-wt \leq \theta \leq 0$ for A_1 and $0 \leq \theta \leq wt$ for A_2. The points on these boundaries correspond to the minimal-time Dubins paths [23] of maximal-rate turn followed by driving straight (black and blue dashed lines in Figure 2-1). Similarly, $B_{1,2}$ are

$$B_1(\psi, t) = \begin{bmatrix} -\rho(2\cos\psi - 1 - \cos(2\psi - wt)) \\ \rho(-2\sin\psi - \sin(-2\psi - wt)) \end{bmatrix} \tag{2.3}$$

$$B_2(\psi, t) = \begin{bmatrix} \rho(2\cos\psi - 1 - \cos(2\psi - wt)) \\ \rho(2\sin\psi - \sin(2\psi - wt)) \end{bmatrix}, \tag{2.4}$$

with $-\psi^\star \leq \psi \leq 0$ for B_1 and $0 \leq \psi \leq \psi^\star$ for B_2. $\psi^\star$ is found by solving

$$2\cos\psi^\star - 1 - \cos(2\psi^\star - wt) = 0.$$

The points on these boundaries correspond to paths of maximal-rate turn in one direction followed by maximal-rate turn in the other direction (red and magenta dashed lines in Figure 2-1).

Given the initial position and orientation of a moving obstacle that follows the described dynamics, its location at any given future time lies within these bounds.

2.1.2 Simplified Collision Region

Equations (2.1)–(2.4) describe the possible locations of the obstacle for a given time t. This region is grown by the *collision radius* r to find the set of locations at which the robot could be in contact with the obstacle at time t. For example, with a circular robot of radius r_1 and a circular obstacle of radius r_2, $r = r_1 + r_2$. As long as the

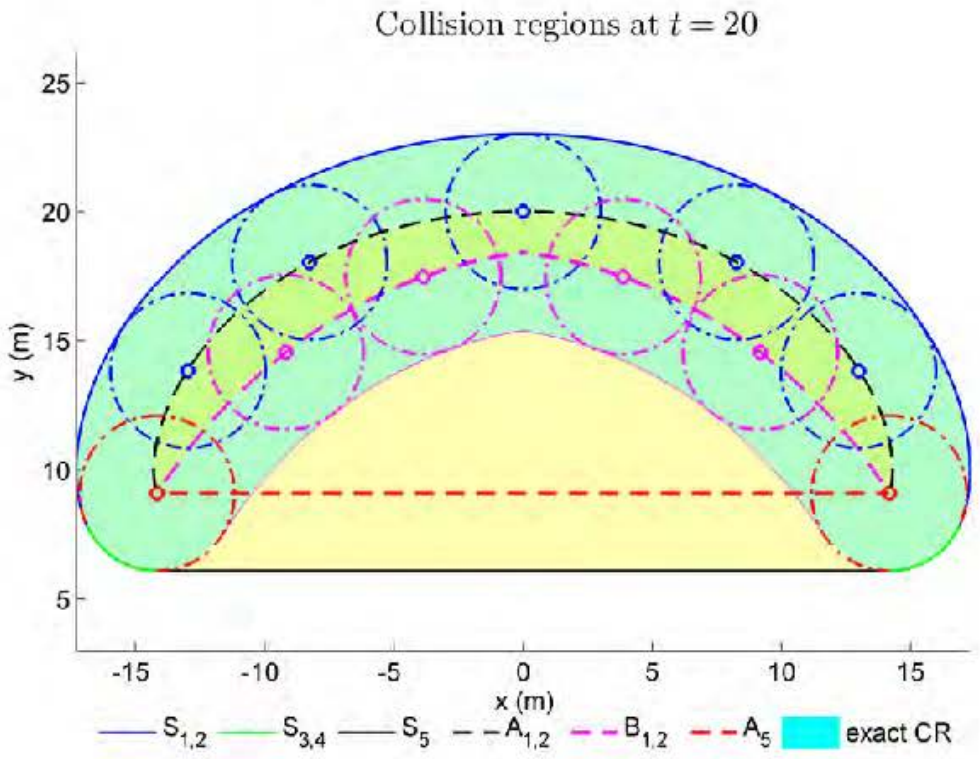

Figure 2-2: Example simplified collision region. The region from Figure 2-1 is grown by the collision radius to produce CR(t), in light blue. CR(t) is then expanded using S_5 into SCR(t), in light yellow, for simplicity.

robot is outside of this grown *collision region* CR(t) (light blue in Figure 2-2) at time t, there cannot be a collision with the obstacle at the specified instant in time.

To simplify the representation of the various segments bounding the collision region, segment A_5 (dashed red line in Figure 2-2) is introduced to replace $B_{1,2}(\psi, t)$ (dashed magenta line). This expands the original reachable set to produce a new region that contains the original set. (See Section 2.1.3 for discussion of why this is a safe simplification). Expanding this simplified reachable set by collision radius r produces the *simplified collision region* (SCR), with boundaries $S_{1,\ldots,5}$. This region is defined for each obstacle at a specific time, written $\text{SCR}_i(t)$, where i indexes the obstacles.

Figure 2-2 shows that the boundary of the simplified collision region for a given instant in time is a combination of up to 5 possible segments, $S_{1,2,3,4}(\theta, t)$ and $S_5(t)$.

Four of these segments are parameterized in θ, which is the direction of the normal vector as measured from the y axis, ranging from $-\pi$ to π. That is, at any point on $S_{1,2,3,4}(\theta, t)$, the normal unit vector $\hat{n}$ is given by

$$\hat{n}(\theta) = \begin{bmatrix} \sin\theta \\ \cos\theta \end{bmatrix}. \tag{2.5}$$

This equation is easily confirmed by checking that the normal vector is orthogonal to the tangent:

$$\hat{n}(\theta) \cdot \frac{\partial S_j}{\partial \theta} = 0 \qquad j = 1, \ldots, 4. \tag{2.6}$$

$S_{1,2}(\theta, t)$ (solid blue in Figure 2-2) are derived from $A_{1,2}(\theta, t)$ by adding r to Equation 2.1 and Equation 2.2 along the outward (with respect to the interior of the reachable set) normal. They are parametrically represented as

$$S_1(\theta, t) = \begin{bmatrix} -\rho(1-\cos\theta) + (vt + \rho\theta + r)\sin\theta \\ -\rho\sin\theta + (vt + \rho\theta + r)\cos\theta \end{bmatrix}, \tag{2.7}$$

$$S_2(\theta, t) = \begin{bmatrix} \rho(1-\cos\theta) + (vt - \rho\theta + r)\sin\theta \\ \rho\sin\theta + (vt - \rho\theta + r)\cos\theta \end{bmatrix}, \tag{2.8}$$

with respective domains

$$S_1: \quad \max(-wt, -\pi) \leq \theta \leq 0, \tag{2.9}$$

$$S_2: \quad 0 \leq \theta \leq \min(wt, \pi). \tag{2.10}$$

$S_{3,4}(\theta, t)$ (solid green in Figure 2-2) are circular arcs of radius r that wrap around the left and right bottom corners of the reachable set. The centers of these arcs are found by substituting $\max(-\pi, -wt)$ and $\max(\pi, wt)$ into Equation 2.1 and Equation 2.2 respectively. If $|wt| \geq \pi$, then it is clear that S_1, S_2, and S_5 enclose the simplified collision region. This is both geometrically intuitive (Figure 2-2) and rigorously addressed (Section 2.1.3). When $|wt| < \pi$, the parametric expressions for these

arcs are

$$S_3(\theta, t) = \begin{bmatrix} -\rho(1 - \cos(\omega t)) + r\sin\theta \\ \rho\sin(\omega t) + r\cos\theta \end{bmatrix}, \tag{2.11}$$

$$S_4(\theta, t) = \begin{bmatrix} \rho(1 - \cos(\omega t)) + r\sin\theta \\ \rho\sin(\omega t) + r\cos\theta \end{bmatrix}, \tag{2.12}$$

with respective domains

$$S_3 : \quad -\pi \leq \theta \leq -\omega t, \tag{2.13}$$

$$S_4 : \quad \omega t \leq \theta \leq \pi. \tag{2.14}$$

Note that for $t > \pi/\omega$, $S_{3,4}$ are no longer part of the boundary of $\mathrm{SCR}_i(t)$.

Finally, $S_5(t)$ is the horizontal line segment joining the lowest points in $S_{1,2,3,4}(\theta, t)$. These are either $S_3(-\pi, t)$ and $S_4(\pi, t)$, or $S_1(-\pi, t)$ and $S_2(\pi, t)$, depending on which pair of curves is defined for $\theta = \pm\pi$ at time t. For this segment, the outward normal unit vector is $[0 \ -1]^T$.

$S_{1,2,3,4}(\theta, t)$ can be combined into a piecewise curve $R(\theta, t)$ for which $\theta \in [-\pi, \pi]$. The region below R and above S_5 defines the simplified collision region $\mathrm{SCR}_i(t)$ for obstacle i at any time t. For clarity, specific segments $S_i(\theta, t)$ will be referred to instead of $R(\theta, t)$ as a whole.

2.1.3 Validity of Using SCR(t) in Place of CR(t)

Guaranteed collision avoidance is maintained when replacing the collision region $\mathrm{CR}(t)$ with $\mathrm{SCR}(t)$, if $\mathrm{SCR}(t)$ includes all of $CR(t)$. Since $S_5(t)$ is the new lower boundary of the region, this inclusion is true if no points on the original boundary of $\mathrm{CR}(t)$ lie below $S_5(t)$. Equivalently, one must show that defining $A_5(t)$ to join $A_1(-\omega t, t)$ and $A_2(\omega t, t)$ as the new lower boundary does not exclude points from the original region, i.e., no points in the replaced boundary segments $A_{\{1,2\}}, B_{\{1,2\}}$ lie

below $A_5(t)$ (see figure 2-3). That is,

$$A_{\{1,2\},y}(\theta, t) \geq A_{5,y}(t) \quad \forall |\theta| > \pi \tag{2.15}$$

and

$$B_{\{1,2\},y}(\psi, t) \geq A_{5,y}(t) \quad \forall |\psi| < \psi^\star. \tag{2.16}$$

Equation 2.15 is only a concern for $\omega t > \pi$ (because $\theta \leq \omega t$), and within this domain, $A_{5,y}(\theta, t) = A_{2,y}(\pi, t)$. Using the first and second derivatives of $A_{2,y}(\theta, t)$ (see Equation 2.2) with respect to t, it is easy to show that $A_{2,y}$ has local minimums at $\theta = \{\pi, 3\pi, \ldots\}$. Equation 2.2 evaluated at these points yields

$$A_{2,y}(\theta, t) = -(vt - \rho\theta),$$

which increases with θ. Clearly, the global minimum occurs at $\theta = \pi$. Hence, Equation 2.15 is satisfied.

Starting with Equation 2.4,

$$\begin{aligned} B_{2,y}(\psi, t) &= \rho(2\sin\psi - \sin(2\psi - \omega t)), \\ \frac{dB_{2,y}}{d\psi} &= \rho(2\cos\psi - 2\cos(2\psi - \omega t)) \\ &= B_{2,x}(\psi, t) + (1 - \cos(2\psi - \omega t)) \\ &\geq B_{2,x}(\psi, t). \end{aligned}$$

From Section 3.2 in Ref. [4], $B_{2,x}(\psi, t) \geq 0$ for the entire domain of ψ. Therefore,

$$\frac{dB_{2,y}}{d\psi} \geq 0 \quad \forall \psi \in [0, \psi^\star].$$

Hence, B_y is an increasing function of ψ, and since and $A_{5,y}(t) = B_{2,y}(0, t)$, Equation 2.16 is satisfied.

Therefore, the modified reachability set is a valid over-bound of the exact reachability set, so the corresponding collision region SCR(t) is a safe over-bound of the

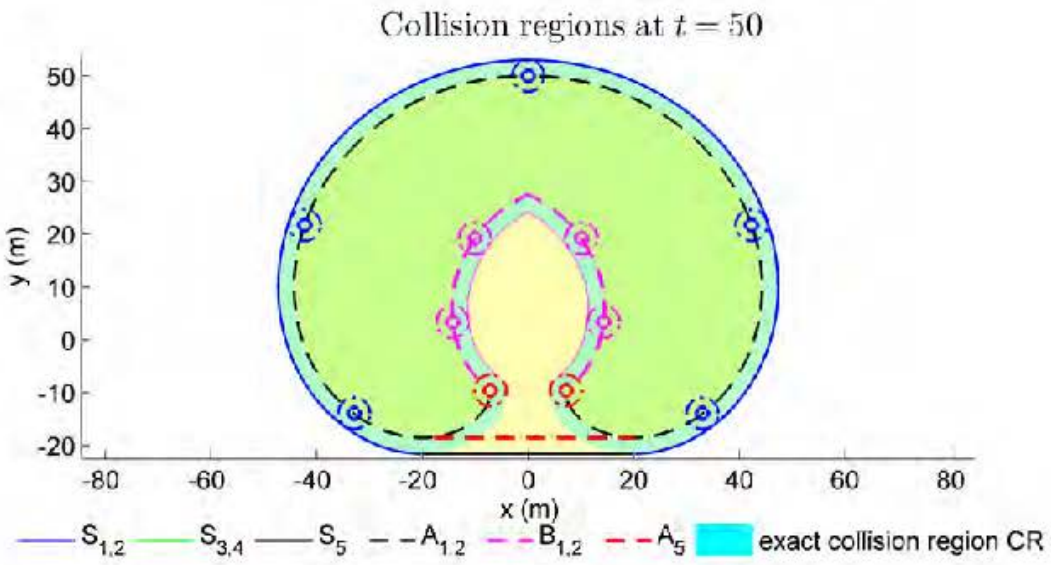

Figure 2-3: SCR($t = 50$). SCR(t) defined using S_5 includes all of the original exact collision region if the dotted red line A_5 lies below $A_{1,2}$ as it curves inward (as time is extended) and all parts of $B_{1,2}$. The proof presented in Section 2.1.3 first shows that, even as t is increased to infinity and the curves $A_{1,2}$ are extended, while they periodically loop back down towards A_5, the low point of each pass is higher than the previous. It is then shown $B_{1,2}$ increase in y away from the endpoints that joins with $A_{1,2}$, hence A_5 also lies below all of $B_{1,2}$.

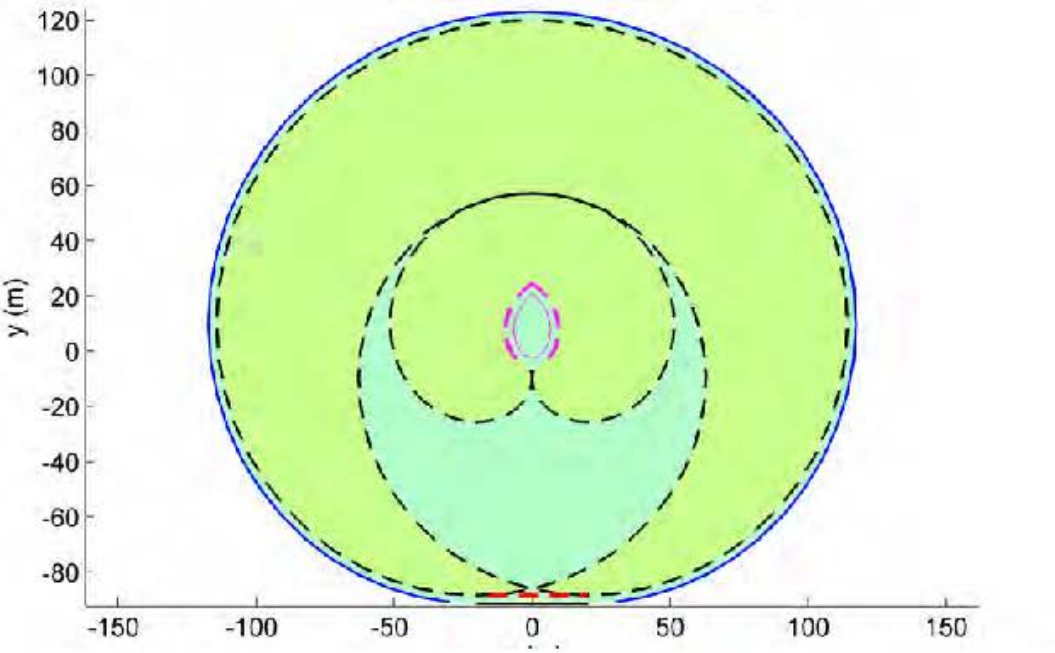

Figure 2-4: SCR($t = 120$). As t is increased, the curves $A_{1,2}$ continue indefinitely, but they can be safely truncated using A_5 since they lie completely above A_5.

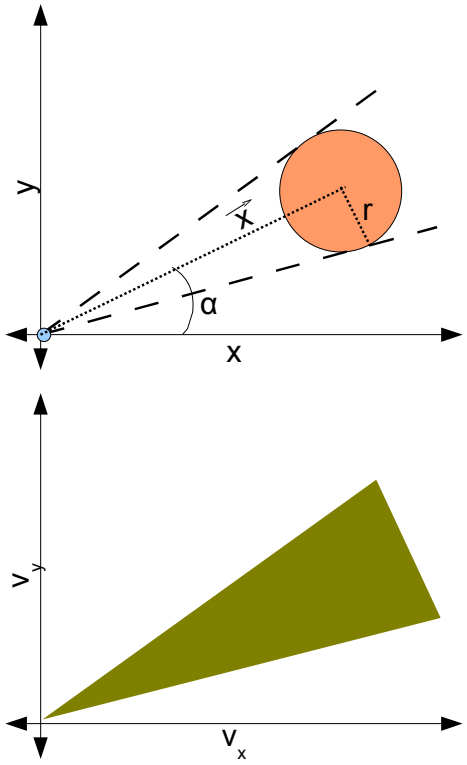

Figure 2-5: Velocity Obstacle associated with a circular, stationary obstacle (red circle in top frame). Should the host robot (blue point at the origin of the top frame) move from the origin with any velocity that lies within the velocity obstacle (green cone in bottom frame), it will eventually collide with the obstacle at some future time.

exact collision region CR(t).

2.2 Velocity Obstacles

This section review the conversion of constraints in physical space to constraints in *velocity space*. The authors of [5] introduce the concept of *velocity obstacles*.

2.2.1 Basic Picture

Consider an obstacle of radius r at initial location $\vec{x}$. Let the host robot be a point mass starting at the origin. If the robot drives indefinitely in the direction of this obstacle, it will collide with the obstacle at some future time. That is, if the angle

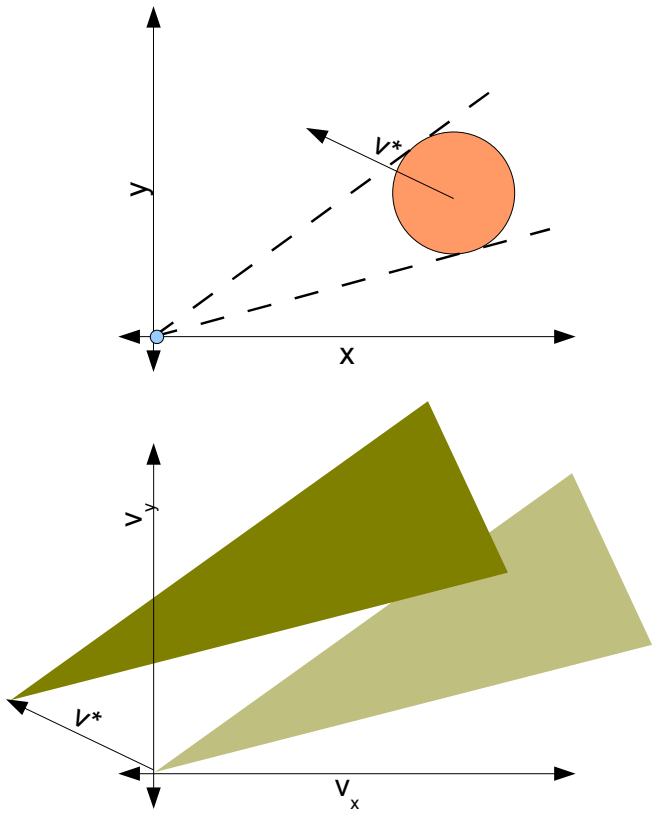

Figure 2-6: Velocity Obstacle associated with a circular obstacle moving along a single-velocity trajectory. Translating the original velocity obstacle by this relative velocity yields the new velocity obstacle. Any velocity from within the green region will eventually result in contact with the moving obstacle.

measured from the positive x axis of a single-direction trajectory lies between $\alpha(\vec{x}) \pm \arctan(r/|\vec{x}|)$, this trajectory will result in a collision. These angular bounds translate directly into a cone in velocity space (green region in Figure 2-5) whose sides are rays pointing along these critical angles and whose base extends out to infinity. A velocity within these bounds (or, a velocity from *within this region of velocity space*) will contact the obstacle if continued indefinitely, while all other velocities are *safe*, as long as the velocity is never changed.

For an obstacle that moves with constant velocity $v^\star$, the same results apply as *relative velocities* (see Figure 2-6). That is, a collision will occur if and only if the relative velocity between the host robot and the obstacle lies within the original

velocity region [24]. Therefore, the original region in velocity space can be simply translated by $v^\star$, and this produces the *linear velocity obstacle*, which applies to obstacles moving with constant velocities. It is also possible to compute the *nonlinear velocity obstacle* for obstacles moving along arbitrary trajectories (See Figure 2-8); the process is discussed in detail in [25].

2.2.2 Equivalent Interpretation

The following introduces an alternate derivation. The results of this approach are identical to the more intuitive case from 2.2.1, but this process is more easily adapted to compute *velocity obstacle sets* (Chapter 3) for unpredictable obstacles, which is the main result of this thesis.

Given a set of coordinates $X(t = \tau)$ that must not be entered at time τ (collision region of a physical obstacle at a given time), there is a corresponding set of velocities

$$V(t = \tau) = \frac{X(\tau)}{\tau}$$

found by simply dividing X by τ. If the vehicle were to start at the origin at $t = 0$ and take a linear, single-velocity trajectory using any of the velocities in the set $V(\tau)$, it would end up at a coordinate inside set X at time τ. Any other single-velocity trajectory would successfully stay clear of this constraint. This set $V(t)$ of velocities to avoid, associated with collisions occurring at a specific future time, is an *instantaneous velocity obstacle* (blue circles in Figure 2-7 and red circles in 2-8).

For moving and stationary obstacles that exist for longer than a single time instant, the host vehicle must jointly avoid all the collision regions at their matching time instants, as given by the trajectory of the obstacle. The union of these resulting constraints for each obstacle forms a 2D shape in velocity space. This defines the velocity obstacle

$$VO(t_0, t_f) = \bigcup V(t) \quad \forall t \in [t_0, t_f]$$

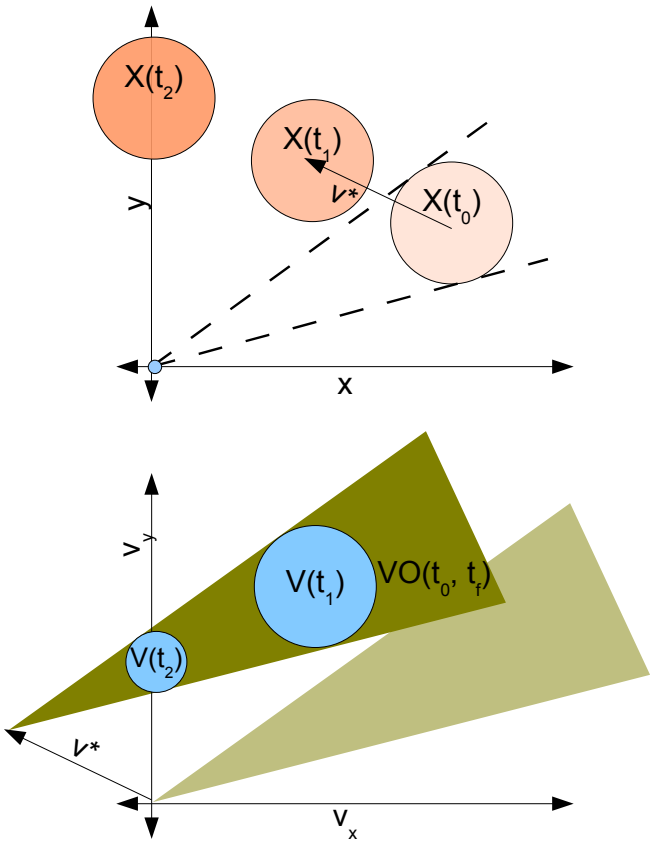

Figure 2-7: Equivalent interpretation of the velocity obstacle for moving obstacle. Instantaneous locations (shades of red in top frame) of the obstacle are converted into velocity-space regions (blue circles in bottom frame) by dividing by time. The union of these regions over a time interval produce (a section of) the same velocity obstacle (green cone) described in section 2.2.1. Note that t_0 must be > 0.

where $t_0 > 0$ and $t_f > t_0$ bound the time window of interest. Single-velocity trajectories using velocities from inside this velocity obstacle would intersect the constrained regions at some time within the specified window. If the obstacle is moving with fixed velocity over the duration of interest, this results in the linear velocity obstacle (see Figure 2-7). Ref. [5] shows that the linear velocity obstacle is a truncated segment of a cone that expands from the origin towards the moving object.

Where the truncation occurs is determined by the time window. It does not make sense to define $V(0)$, because contact with the obstacle at $t = 0$ is simply determined by the initial conditions. As the lower time bound approaches 0, the open end of the

cone tends towards infinity, as collisions in the very near future would entail driving towards the obstacle at high speeds. When these speeds exceed dynamic capacities of the host vehicle, there are effectively no longer of practical interest, and the velocity obstacle can be truncated at some small initial time. As the upper time bound approaches infinity, the sharp end of the cone converges to the obstacle's terminal velocity (see figure 2-7: as t increases, $V(t)$ shrinks and tends towards the tip of the VO cone).

Nonlinear velocity obstacles can be computed for objects that move in arbitrary ways over the time window. Layering the velocity constraints generated at each time instant produces a warped cone (see Figure 2-8), and the calculation and representation of this shape is discussed in Ref. [25]. Velocities outside of this region generate safe, single-velocity trajectories; the safety of nonlinear or speed-varying *robot* trajectories are explored in [26].

The standard approach for defining and using velocity obstacles in the existing literature ([5, 15, 16, 24, 27]) requires a mapping that gives the location of the object as a function of time over the specified window in order to produce the velocity obstacle. This typically limits the horizon to accurately predictable time-scales and requires any prediction error to be handled by growing the collision size. While there has been extensive work in improving prediction of obstacle motion [28, 29], this framework is nonetheless generally ill-suited for guaranteeing long term collision avoidance. This thesis extends the velocity obstacle concept to scenarios in which the predicted trajectory is unavailable.

2.3 Invariant Safe States

This section briefly reviews the concept of *invariant safe states*, which is often used to prove extended collision avoidance. This material will be relevant to the guarantees of collision avoidance (Chapter 4) and to properties of iterative planners (Chapter 5).

In extended horizon scenarios, it is often only possible to make detailed motion plans into the near future without explicitly reaching a terminal goal state. The

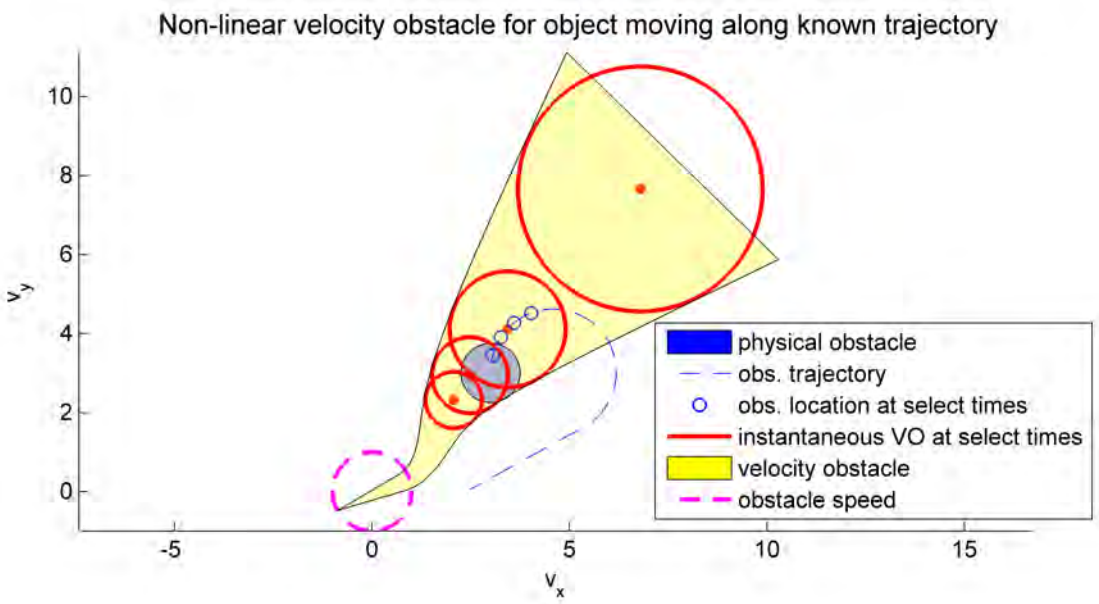

Figure 2-8: The nonlinear velocity obstacle for an object moving along a *known* curved trajectory. Note that the physical obstacle and its path are sketched and super-imposed upon the velocity space coordinates. This image is only for visualization; these entities are defined in physical space (x and y, as opposed to v_x and v_y), and have no actual meaning in this frame. If the obstacle trajectory is assumed to continue indefinitely at some fixed velocity, the tip of the nonlinear velocity obstacle terminates at the coordinate corresponding to that velocity, and if the obstacle were to drive in circles, the tip of the nonlinear velocity obstacle converges to the origin; this is the behavior of the host vehicle that collides with the obstacle in the very distant future.

final goal may be far out of reach or the environment may be too dynamic, so a short term plan is executed, bringing the robot closer to the goal with the intention of regenerating more short-term plans in the future. Typically, it is fairly straightforward to avoid collisions within the planning horizon. However, it is often difficult to account for issues that may become unavoidable farther in the future. That is, even with the ability to re-create a new short-term plan before the horizon of the current plan expires, decisions from the current plan, combined with the unpredictability of the environment, may make it impossible to find a safe trajectory at the next planning iteration. The momentum of the host vehicle may make an evasive maneuver impossible, or the terminal position of the host robot may be in the path of a moving obstacle.

To cleanly handle such scenarios, many iterative path planners (such as the ones in [1] and [2]), define *invariant safe states*. This is the set of host vehicle states such that

- all safety (collision avoidance) constraints are satisfied,
- it is dynamically feasible to transition back into itself.

For example, in static environments, staying motionless without contacting a static obstacle would be an invariant safe state. For autonomous vehicles driving in an urban setting [1], the rules of the road make stopping in certain areas, like beside a curb, invariant safe states. Such a state can be maintained indefinitely if need be. This means that if a finite-horizon plan terminates in such a state, the plan can be trivially extended for any amount of time with guaranteed collision avoidance. Furthermore, any state that can transition into an invariant safe state can also be deemed safe and considered a valid terminal condition for finite-trajectory plans.

Therefore, if an iterative planner is limited to choosing finite-horizon plans that terminate in safe states, such a planner is guaranteed to avoid collisions for arbitrary time scales. The existence of at least one feasible option every time a new plan is computed is ensured, as the planner can always choose to transition into the invariant safe state and stay there. However, nothing can be said about the existence of a solution that brings the robot to the desired goal; enforcing this constraint may in fact cause the planner to overlook potential trajectories to the goal in favor of trajectories that cling in proximity to the invariant safe states. This issue must be dealt with on a case to case basis, depending on the specific goals and challenges of each application.

In relatively unstructured environments, it can be very challenging to define invariant safe states. The scenario at the core of this thesis involves multiple dynamic obstacles whose trajectories are unpredictable, governed only by their known dynamical limitations. The classic invariant safe states of bringing the host vehicle to a stop at some location no longer apply, as it could still be possible for the host vehicle to be struck by another vehicle. It will be shown that the velocity obstacle sets derived

in Chapter 3 can be used to define invariant safe states, thereby allowing guaranteed infinite horizon collision avoidance.

2.3.1 Inevitable Collision States

A popular variant approach to guaranteed safety uses *Inevitable Collision States* [17, 30]. Using the dynamics of the vehicle and the model of the environment, it is possible to identify vehicle states from which any control input would lead to a collision. If all such states can be exhaustively found, then the host vehicle need only avoid these states to guarantee collision avoidance.

This is closely related to the use of invariant safe states: maintaining access to a safe state guarantees safety; alternatively, never entering a restricted inevitable collision state achieves the same goal. Depending on the scenario, it may be more sensible to satisfy one of these conditions instead of the other. If the inevitable collision states are easily found, it is straightforward to simply check that the commanded input does not immediately enter such a state. However, in the case of multiple obstacles, inevitable collision states include surrounded, as well as soon-to-be surrounded, configurations. Solving exactly for the collisions that lead to unavoidable collisions is a very difficult problem whose solution may be more naturally expressed in terms of invariant safe states.

Chapter 3

Velocity Obstacle Set Definition

The objective of this work is to find conditions that guarantee collision avoidance, even without exact knowledge of the obstacles' future locations as functions of time. This is achieved by mapping the collision regions of the reachable sets of the obstacles (Section 2.1) into regions in velocity space and using the velocity obstacle concept (Section 2.2) to produce a concise representation of velocities at which the host vehicle can travel without ever intersecting the collision region of an obstacle for any time. By keeping clear of the reachable sets of each obstacle, the host vehicle avoids any possible collisions without explicitly anticipating trajectories for the obstacles.

As described in Section 1.2, the obstacles are assumed to follow unicycle dynamics: they have a fixed forward speed v and a limited angular acceleration of $|\dot{\theta}| \leq \omega$, this giving them a minimum turning radius of $\rho = \frac{v}{\omega}$. In addition to v and ω, the collision radius parameter r and the current locations and headings of each obstacle are used to calculate these velocity space constraints.

3.1 Definition

Given the speed and turn-rate constraint of any unicycle model obstacle, one can compute an over-bound $\mathrm{SCR}_i(t)$ of its dynamically feasible collision set as a function of time (Section 2.1.2). At every instant in time, one can divide $\mathrm{SCR}_i(t)$ by the time to extract a corresponding region in velocity space, the *simplified velocity region*

(SVR), expressed as

$$\mathrm{SVR}_i(t) = \frac{\mathrm{SCR}_i(t)}{t} \tag{3.1}$$

for any $t > 0$. Any point mass (that starts at the origin) traveling with velocity inside $\mathrm{SVR}_i(t)$ will enter the collision region, whereas a point mass with a velocity outside of this set will remain clear of $\mathrm{SCR}_i(t)$.

The boundary $C_i(t)$ of $\mathrm{SVR}_i(t)$ consists of the segments

$$Q_{1,2,3,4}(\theta, t) = \frac{1}{t} S_{1,2,3,4}(\theta, t) \tag{3.2}$$

$$Q_5(t) = \frac{1}{t} S_5(t), \tag{3.3}$$

using Equation 2.7-Equation 2.14. Any input outside of this region in velocity space will return a linear, single-speed trajectory that will be safe at the given instant. For safety over a window of time, one needs to layer all the instantaneous constraints together. The union of these constrained regions in velocity space is defined as the *velocity obstacle set* (VOS) (Figures 3-1 and 3-2), which thus includes all of the single-velocity trajectories that could possibly generate a collision within the time window. For the i^{th} obstacle,

$$\mathrm{VOS}_i(t_0, t_f) = \bigcup \mathrm{SVR}_i(t) = \bigcup \frac{\mathrm{SCR}_i(t)}{t}, \ \forall t \in [t_0, t_f]. \tag{3.4}$$

The VOS is a nonlinear velocity obstacle computed using the entire reachability set of a dynamic object, instead of just a single estimated trajectory as in prior work [25]. Velocities outside of the VOS give a linear trajectory that is guaranteed safe with respect to all dynamically feasible trajectories of the obstacle for the duration of the time window.

Equivalently, the VOS can be interpreted as the union of all the classic nonlinear, single-trajectory velocity obstacles (see Figure 2-8 for the classic nonlinear velocity obstacle) associated with each of the dynamically feasible obstacle trajectories, defined over the same time window (see Figures 3-3 to 3-5). These two entities would

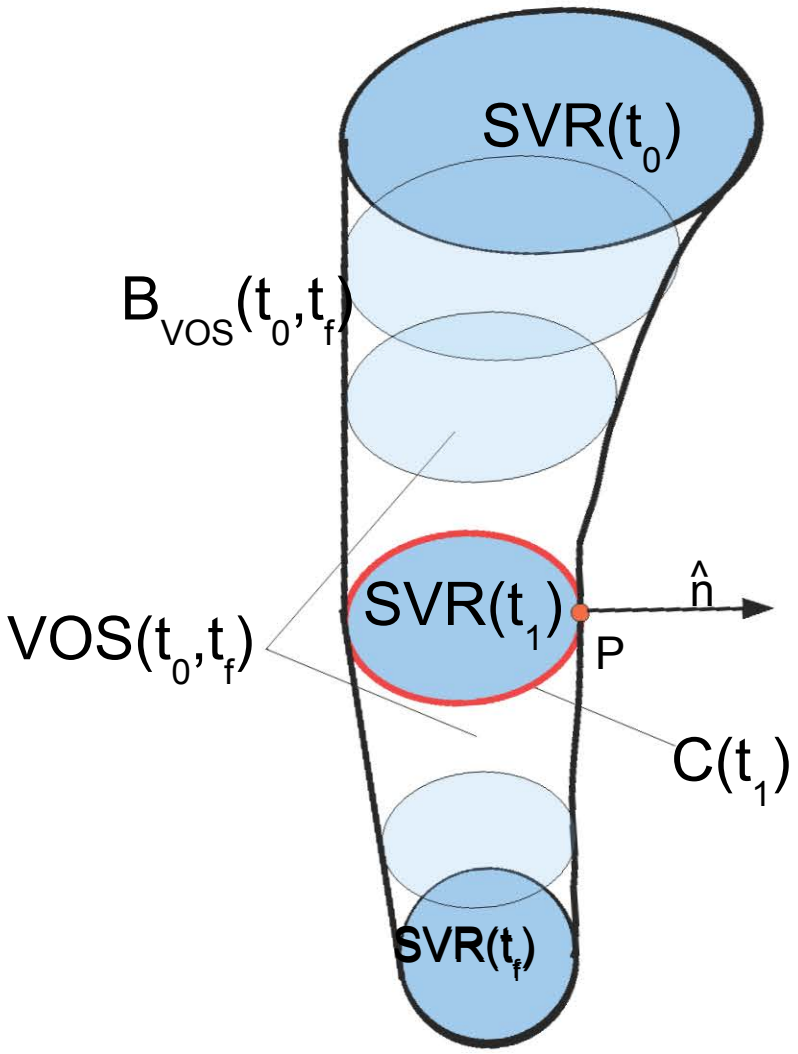

Figure 3-1: Conceptual sketch of example velocity obstacle set (region in velocity space). $\mathrm{VOS}(t_0, t_f)$ is the union of $\mathrm{SVR}(t)$ for $t \in [t_0, t_f]$. The boundary of each $\mathrm{SVR}(t)$ is $C(t)$. $C(t)$ is composed of segments $Q_{1,2,3,4}(\theta, t)$ and $Q_5(t)$. The boundary of VOS is B_{VOS}. A point P on the boundary has an associated normal vector $\hat{n}$.

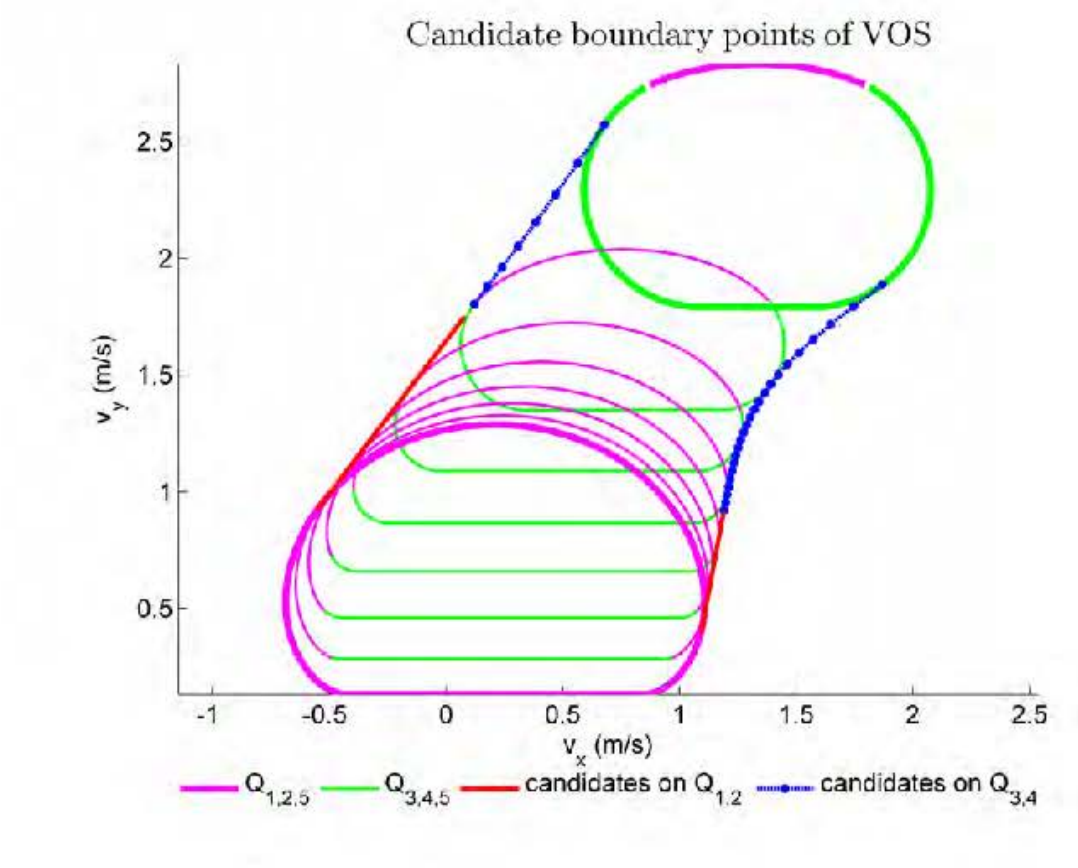

Figure 3-2: Computed example velocity obstacle set with parameters $x_0 = 4$, $y_0 = 4$, $\rho = 6.063$, $r = 1.5$, $v = 1$. $C(t_0)$ and $C(t_f)$ are plotted thicker than $C(t)$ for intermediate times (Each C is plotted as its components $Q_{1,2,3,4,5}$). These "terminal regions" boundaries $C(t_0)$ and $C(t_f)$ along with candidate points (in blue and red) found on $Q_{1,2,3,4,5}(t)$ for intermediate $t \in (t_0, t_f)$ make up the boundary of VOS(t_0, t_f).

be exactly equivalent, as they both return the set of velocities that may put the robot within contact distance of an obstacle location at some future time. The only distinction to be made is that, as the VOS is defined using the *simplified* region SCR, the VOS is actually a superset of the union of all feasible velocity obstacles, i.e.,

$$\bigcup VO|_{\text{traj}}(t_0, t_f) \subset VOS(t_0, t_f)$$

for all feasible trajectories. It will be shown in section 3.2.4 that the two are in fact very nearly the same, and the simplification typically does not affect the final result (they are in fact identical if the final boundary of the VOS does not contain elements of Q_5).

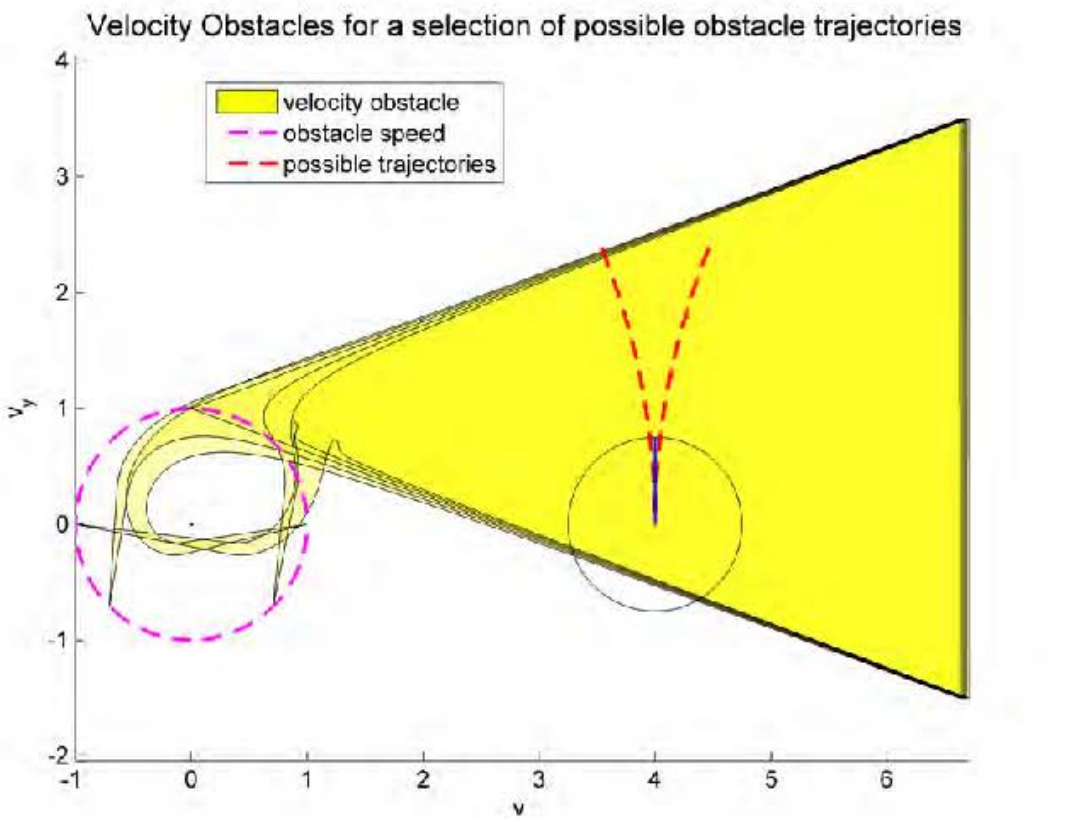

Figure 3-3: Velocity obstacles for different possible trajectories of an obstacle with $x_0 = 4$, $y_0 = 0$, $\rho = 6.063$, $r = 1.5$, $v = 1$. Jointly avoiding all possible velocity obstacles would guarantee safety. Exhaustively simulating all these trajectories out to infinity is not a good solution.

As discussed in the following sections, using the reachability sets as a function of time is a tractable way to compute the VOS. On the other hand, it would not be computationally feasible to arrive at the same theoretical result by simulating all possible physical trajectories over an infinite time horizon, computing the associated velocity obstacles, and then taking the union of the results (Figure 3-3). Ultimately, the desired result is a representation of the boundary of the VOS.

3.2 Conditions for Boundary Points of the VOS

This section derives Algorithm 1 for finding the boundary $B_{\mathrm{VOS}_i}(t_0, t_f)$ of the region $\mathrm{VOS}_i(t_0, t_f)$, where $t_0 > 0$ and $t_f \leq \infty$ (See Section 3.3.2 for a detailed discussion of these limits). All candidate points from the curves $C_i(t)$ whose time derivatives

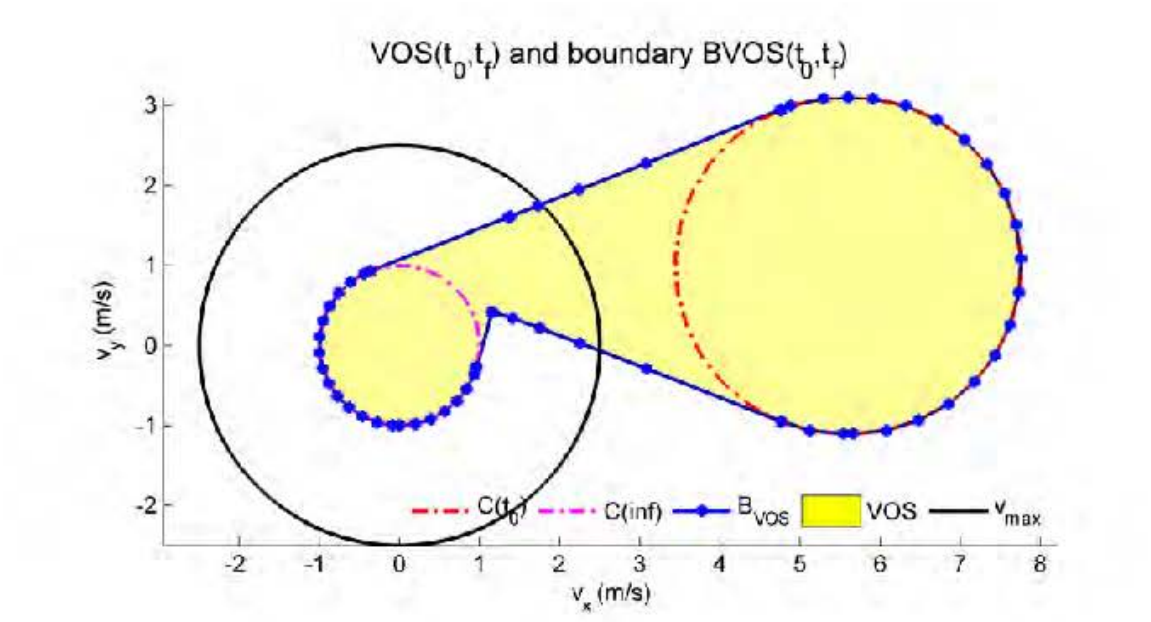

Figure 3-4: Velocity Obstacle Set for the same conditions. This set is computed using reachability. It is always contains the region in Figure 3-3 as a subset; in this case they are identical if the region in Figure 3-3 were exhaustively defined.

are perpendicular to the normal vector are found. These points define the points of tangency between $\text{SVR}_i(t)$ and $\text{SVR}_i(t+\delta t)$ (see Figure 3-1). The set of all points satisfying this condition contains the boundary $B_{\text{VOS}_i}(t_0, t_f)$, as summarized in Remark 1. A detailed explanation follows.

Every point P on $B_{\text{VOS}_i}(t_0, t_f)$ must be a boundary point on at least one of the underlying simplified velocity regions $\text{SVR}_i(t)$ for some $t \in [t_0, t_f]$ (Figure 3-1). This follows trivially from the fact that P must be found in some $\text{SVR}_i(t)$ by definition of VOS_i in Equation 3.4, and that if P were on the interior instead of the boundary $C_i(t)$ of $\text{SVR}_i(t)$, some neighboring point of P would lie outside of $B_{\text{VOS}_i}|_P$.

With the exception of corners, every point P on the curve $B_{\text{VOS}_i}(t_0, t_f)$ has a *local outward normal vector* $\hat{n}|_P$. See Figure 3-7 for an example of a corner. Corners in the boundary would have two different normals approaching from either side, and the normal at that point is undefined. These intersections between local boundary segments are dealt with at the last step in this algorithm, but the following conditions applied locally to each segment and its normal are still valid as necessary conditions

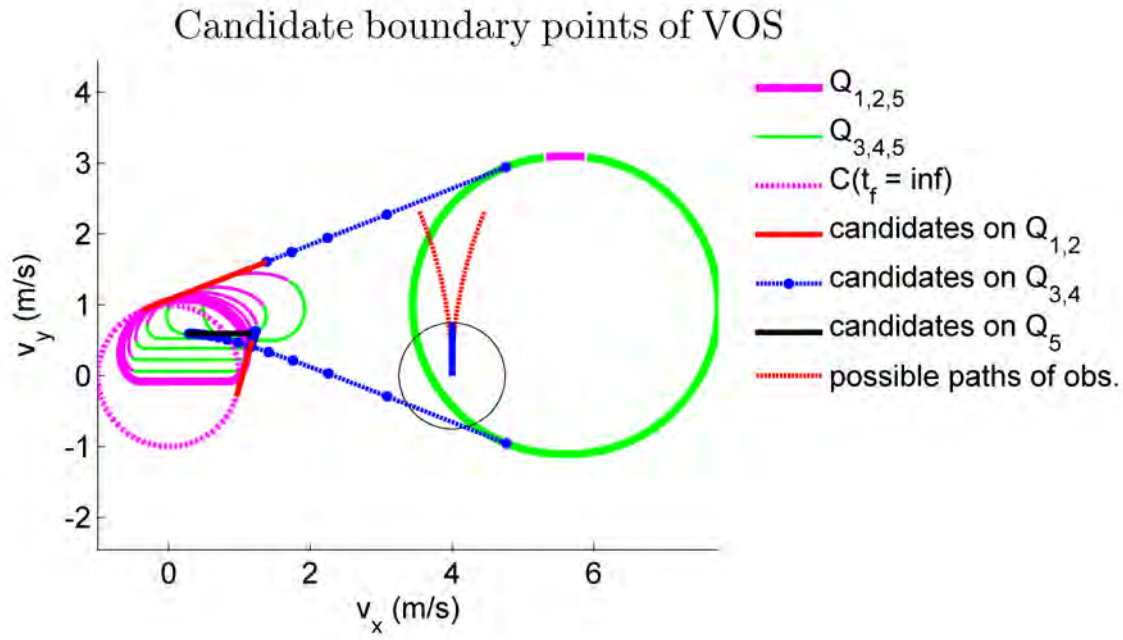

Figure 3-5: This is how the boundary of the VOS is found. Note that in this Figure (as well as Figure 3-3), the physical trajectory is just a reference image.

of boundary points.

By definition of a boundary, there must not be any neighboring points inside $\text{VOS}_i(t_0, t_f)$ that reach farther out in the direction of $\hat{n}|_P$. This condition implies the standard equation for the normal vector

$$\hat{n}|_P \cdot \frac{\partial}{\partial s} B_{\text{VOS}_i}|_P(s) = 0,$$

where s is any variable parameterizing the curve B_{VOS_i}. $B_{\text{VOS}_i}(s \pm \delta s)$ gives relevant candidate neighboring points of P within the set VOS_i that must not be farther out along $\hat{n}|_P$. Similarly, since P is found in some $\text{SVR}_i(t)$ which is in VOS_i, one can also write the necessary condition

$$\hat{n}|_P \cdot \frac{\partial}{\partial s} C_i|_P(s) = 0 \tag{3.5}$$

for any variable s for which $C_i(s)$ – the boundary of SVR_i – is continuously defined such that $C_i(s \pm \delta s) \in \text{VOS}_i(t_0, t_f)$.

C_i is composed of the segments $Q_{1,2,3,4}(\theta, t)$ and $Q_5(t)$ from Equation 3.2 - Equation 3.3, and the parametric expressions for these segments can each take the place of C_i in Equation 3.5, yielding

$$\hat{n}|_P \cdot \frac{\partial}{\partial s} Q_{i,j}|_P(s) = 0, \tag{3.6}$$

where i indexes the dynamic obstacle, and $j \in \{1, \ldots, 5\}$ as piece-wise segments of C_i.

By choosing $s = \theta$ where θ is the parameter for the segments $Q_{1,2,3,4}(\theta, t)$, Equation 2.6 is recovered (scaled by a factor of $\frac{1}{t}$)

$$\hat{n} \cdot \frac{\partial S_j}{\partial \theta} = 0 \; j = 1, \ldots, 4, \tag{Equation 2.6}$$

the solution of which is Equation 2.5:

$$\hat{n}(\theta) = \begin{bmatrix} \sin\theta \\ \cos\theta \end{bmatrix}. \tag{Equation 2.5}$$

This means that, if P is found on some segment $Q_{1,2,3,4}(\theta, t)$, then the normal vector $\hat{n}$ associated with B_{VOS_i} is the same normal vector associated with $\text{SCR}_i(t)$ and $\text{SVR}_i(t)$. Similarly, $Q_5(t)$ could be generically parameterized by arc-length s, recovering the normal vector

$$\hat{n}|_{P,Q_5(t)} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}. \tag{3.7}$$

Intuitively, these results indicate that the normal vectors associated with the overall boundary $B_{\text{VOS}_i}(t_0, t_f)$ at any point P must match with the normal vectors found locally on the segments $Q_j(t)$ at P (see Figure 3-1). Otherwise, a point on $Q_j(t)$ would lie outside the boundary $B_{\text{VOS}_i}(t_0, t_f)$.

The expression for $\hat{n}|_P$ can then be substituted into the necessary condition (Equa-

tion 3.6 with the time parameter t substituted for s) to obtain

$$\begin{bmatrix} \sin\theta \\ \cos\theta \end{bmatrix} \cdot \frac{\partial}{\partial t} Q_{i,j}|_P(\theta, t) = 0; \quad j \in \{1, \ldots, 4\} \tag{3.8}$$

and

$$\begin{bmatrix} 0 \\ 1 \end{bmatrix} \cdot \frac{\partial}{\partial t} Q_{i,5}|_P(t) = 0 \tag{3.9}$$

as necessary conditions for a point P to be on the boundary $B_{\mathrm{VOS}_i}(t_0, t_f)$, as long as $Q_{i,j}(\theta, t \pm \delta t) \in \mathrm{VOS}_i(t_0, t_f)$. This latter statement is true for all t in the open interval (t_0, t_f). (At the endpoints t_0 and t_f themselves, these conditions need not hold; just like how the global minimum of a function on a closed interval can be found on the end of the interval without the needing the derivative to be 0 there.) This condition drives how the boundary is found.

Remark 1. *Any point P in $B_{VOS_i}(t_0, t_f)$ must either satisfy Equation 3.8 or Equation 3.9, or lie on the curves $C_i(t_0)$ or $C_i(t_f)$. Graphically, Equation 3.8 and Equation 3.9 describe points that are local tangent points between $SVR_i(t)$ and $SVR_i(t+\delta t)$ (Figure 3-1). Finding all such points results in a larger set that contains all possible points in $B_{VOS_i}(t_0, t_f)$. This set also includes local extrema that are not necessarily global boundary points; see Figure 3-6. Algorithm 1 is the result of these conditions.*

3.2.1 Solving the Necessary Conditions

This section show in detail how Equation 3.8 and Equation 3.9 are solved. Without loss of generality, assume the host robot is located at the origin, and the obstacle is located at (x_0, y_0), pointing along the positive y axis (so that the heading is zero). A different initial condition can be first rotated to match this heading, and the corresponding (x_0, y_0) is easily computed, and the final result is rotated back to the original frame (lines 1 and 20 in Algorithm 1).

Algorithm 1: Solving for the boundary of VOS (See figure 3-6)

Input: obstacle$_i$: $\{x_0, y_0, \theta_0, v, \rho\}, t_0, t_f$
Output: B_{VOS_i}

rotate (x_0, y_0) by $-\theta_0$ (vehicle heading);
{candidate segments} $\leftarrow \emptyset$;

// add $C(t_0)$ and $C(t_f)$ to consideration
for $t \in \{t_0, t_f\}$ **do**
    find $C(t)$ using (3.2),(3.3);
    add $C(t)$ to {candidate segments} ;

// add candidate points on $Q_{1,2}$
{critical angles} = solve conditions in (3.13),(3.12);
for $\theta^\star \in \{critical\ angles\}$ **do**
    $t_1 = \max(\frac{\theta^\star}{w}, t_0)$;
    // points make linear segment between $Q_{1,2}(\theta^*, t_1)$ and $Q_{1,2}(\theta^*, t_f)$
    find segments using (3.10);
    add segment to {candidate segments};

// add candidate points on $Q_{3,4}$
{local segments} $\leftarrow \emptyset$;
for $t \in [t_0, \min(\frac{\pi}{\omega}, t_f)]$ **do** // discretize t
    {points} = solve conditions in (3.17),(3.16) using Equation 3.24;
    append {points} onto {local segments};
add {local segments} to {candidate segments} ;

// add candidate points on Q_5
t_h = solve condition in (3.19);
// endpoints of $Q_5(t_h)$ = $Q_{\{2,1\}\text{or}\{4,3\}}(\pm\pi, t_h)$
find $Q_5(t_h)$ using (3.10),(3.15);
add $Q_5(t_h)$ to {candidate segments};

// candidate segments now includes all possible boundary points
boundary = combine {candidate segments};
rotate boundary by θ_0;
return boundary;

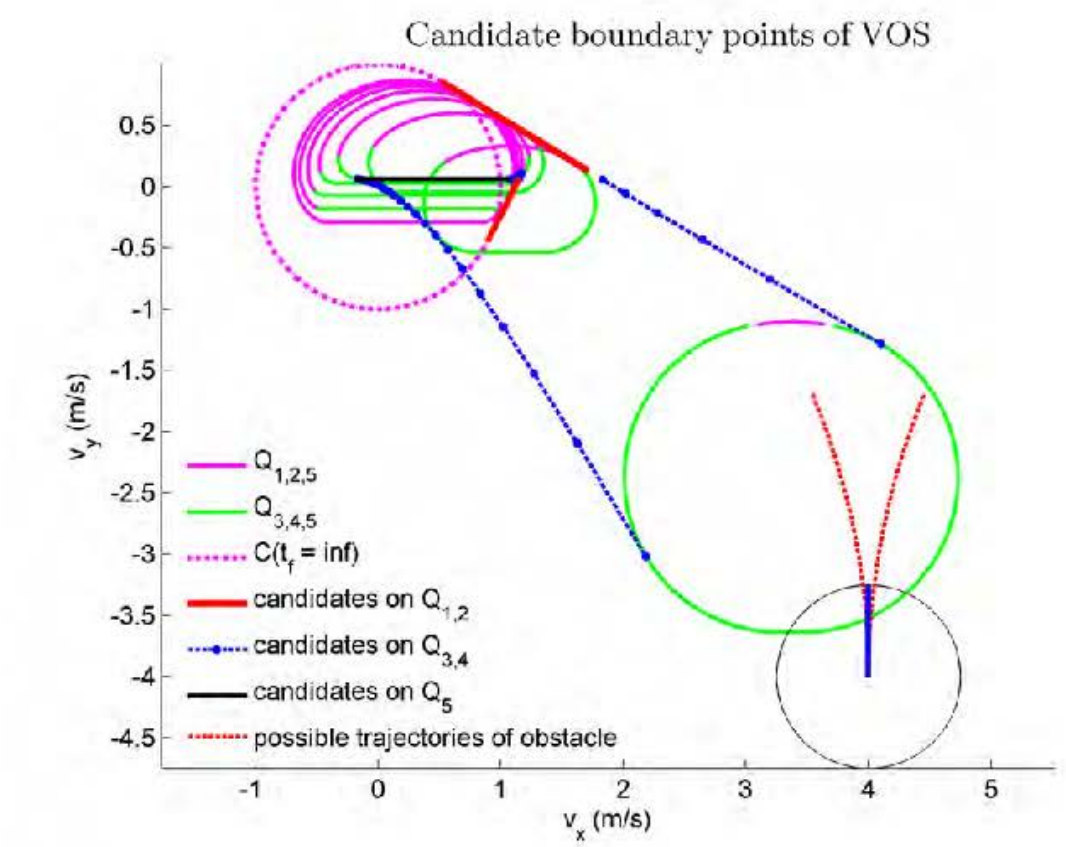

Figure 3-6: Example candidate points in $B_{\text{VOS}}(t_0, t_f)$, using parameters $x_0 = 4$, $y_0 = -4$, $\rho = 6.063$, $r = 1.5$, $v = 1$. Various instances of $C(t)$ are outlined in green and magenta. Candidate boundary points drawn in red, blue, and black. The physical obstacle is sketched for reference, but note that the rest of the plot is in velocity space, and the physical obstacle cannot actually be expressed in these coordinates and needs to be on separate axes, as done technically correctly in Figures 2-5, 2-6, and 2-7.

i) Points on $Q_{1,2}(\theta, t)$:

This corresponds to lines 6–9 in Algorithm 1. First consider points on the segments $Q_{1,2}(\theta, t)$ that may satisfy Equation 3.8. From Equation 3.2 and Equation 2.8,

$$Q_2(\theta, t) = \frac{1}{t} \begin{bmatrix} x_0 + \rho(1 - \cos\theta) + (vt - \rho\theta + r)\sin\theta \\ y_0 + \rho\sin\theta + (vt - \rho\theta + r)\cos\theta \end{bmatrix}. \tag{3.10}$$

The domain for $Q_2(\theta, t)$ is

$$Q_2(\theta, t): \quad t \in [t_0, t_f], \quad \theta \in [0, \min(\omega t, \pi)], \tag{3.11}$$

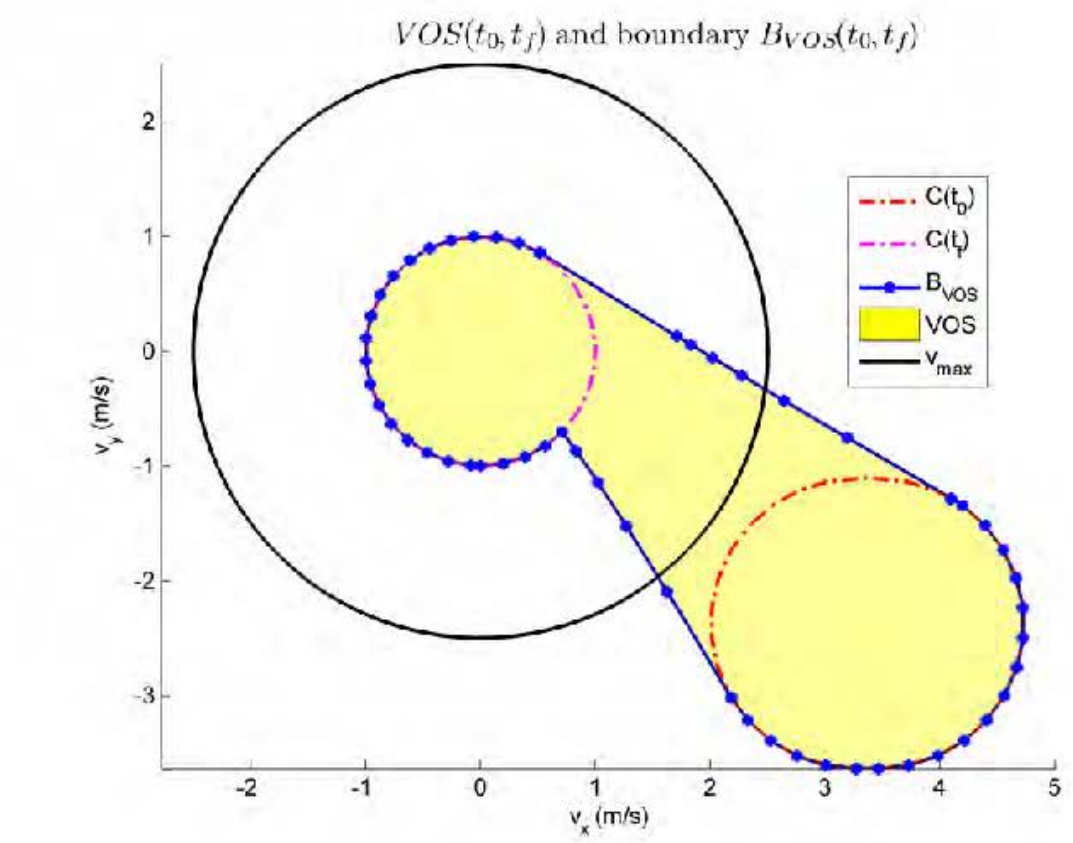

Figure 3-7: Final boundary of VOS from example in figure 3-6. The same parameters of $x_0 = 4$, $y_0 = -4$, $\rho = 6.063$, $r = 1.5$, $v = 1$ are used. $C(t_0 = \epsilon)$ and $C(t_f \to \infty)$ are outlined in red and magenta. B_{VOS} drawn in blue. The maximum host vehicle velocity drawn in black; $C(t_0 = \epsilon)$ lies outside of this circle.

which comes directly from Equation 2.10. Taking the derivative with respect to time and substituting into Equation 3.8 yields

$$\begin{aligned}
\frac{\partial Q_2(\theta,t)}{\partial t} &= -\frac{1}{t^2}\begin{bmatrix}(x_0+\rho)+\rho\cos\theta+(r-\rho\theta)\sin\theta\\ y_0-\rho\sin\theta+(r-\rho\theta)\cos\theta\end{bmatrix},\\
0 &= \frac{\partial}{\partial t}Q_2(\theta,t)\cdot\begin{bmatrix}\sin\theta\\ \cos\theta\end{bmatrix},\\
0 &= -\frac{1}{t^2}\left((x_0+\rho)\sin\theta+y_0\cos\theta+r-\rho\theta\right). \qquad (3.12)
\end{aligned}$$

For any t and θ in the specified domain that satisfies Equation 3.12, $Q_2(\theta, t)$ is a potential boundary point in $B_{\text{VOS}}(t_0, t_f)$.

Identical steps lead to the necessary condition

$$-\frac{1}{t^2}\left((x_0-\rho)\sin\theta + y_0\cos\theta + r + \rho\theta\right) = 0 \tag{3.13}$$

for points of $Q_1(\theta, t)$, defined in the domain

$$Q_1(\theta,t): \quad t \in [t_0, t_f], \quad \theta \in [\max(-\omega t, -\pi), 0].$$

For any combination of t and θ within the domain of Q_1 that satisfies Equation 3.13, $Q_1(\theta, t)$ is a potential boundary point in $B_{\text{VOS}}(t_0, t_f)$.

Note that, excluding $t = \infty$ and assuming $t \neq 0$, Equation 3.12 and Equation 3.13 can be reduced to functions of θ alone. Each has at most two values of $\theta^\star$ that satisfy the equation within its domain, and these values are easily found numerically. Candidate boundary points are given by $Q_j(\theta^\star, t)$ for $t \in [t_1, t_f]$, where $t_1 = \max({}^{\theta^\star}\!/_{\omega}, t_0)$, since for any $t < {}^{\theta^\star}\!/_{\omega}$, $\theta^\star$ would not be in be domain of θ (see Equation 3.11).

For a solved value of $\theta^\star$, the locus of all candidate boundary points is a linear segment between $Q_j(\theta^\star, t_1)$ and $Q_j(\theta^\star, t_f)$ for $j = \{1, 2\}$ (red lines in Figure 3-6). This is because $Q_j(\theta^\star, t)$ Equation 3.10 is a linear function of $\frac{1}{t}$, which is well defined for $t \neq 0$.

Physically, if the dynamic obstacle were turn to at maximum rate ω to $\theta^\star$ and then continue driving directly in this direction, in order to avoid collision, the host robot would need to choose a path that is deflected away from the obstacle with the widest clearance, compared to what it would have to done to avoid any other feasible obstacle trajectory.

ii) Points on $Q_{3,4}(\theta, t)$: This corresponds to lines 11–15 in Algorithm 1.

Next, all possible boundary points on $Q_{3,4}(\theta, t)$ are found. Like segments $S_{3,4}(\theta, t)$ (Equation 2.11-Equation 2.12), the domains of segments $Q_{3,4}(\theta, t)$ are

$$\begin{aligned} Q_3(\theta,t): &\quad t \in [t_0, t_f], \quad \theta \in [-\pi, -\omega t], \\ Q_4(\theta,t): &\quad t \in [t_0, t_f], \quad \theta \in [\pi, \omega t]. \end{aligned} \tag{3.14}$$

Due to the limits on the domain of θ, when solving for $B_{\text{VOS}}(t_0, t_f)$, one only needs to consider points in $Q_{3,4}$ from times in the range $t \in [t_0, \min(\pi/w, t_f)]$. As seen in Figure 3-6, as t increases, $Q_{3,4}$ eventually disappear.

From Equation 3.2 and Equation 2.12,

$$Q_4(\theta, t) = \frac{1}{t}\begin{bmatrix} x_0 + \rho(1 - \cos(\omega t)) + r\sin\theta \\ y_0 + \rho\sin(\omega t) + r\cos\theta \end{bmatrix}, \tag{3.15}$$

$$\frac{\partial Q_4}{\partial t} = \frac{-1}{t^2}\begin{bmatrix} (x_0 + \rho) - \rho\cos(\omega t) - \rho\omega t\sin(\omega t) + r\sin\theta \\ y_0 + \rho\sin(\omega t) - \rho\omega t\cos(\omega t) + r\cos\theta \end{bmatrix}.$$

Given any value of t from within the domain, the necessary condition from Equation 3.8 requires

$$\frac{\partial}{\partial t}Q_4(\theta, t) \cdot \begin{bmatrix} \sin\theta \\ \cos\theta \end{bmatrix} = 0, \tag{3.16}$$

which can be solved numerically for θ. Alternatively, see Section 3.2.3 for an equivalent analytic geometric solution to these conditions. For a given t, Equation 3.16 has between zero and two valid solutions for $\theta \in [0, 2\pi)$ (this is clear from the geometric interpretation). Any of these solutions that fall within the domain $\theta \in [\omega t, \pi]$ is considered a valid candidate boundary point in $B_{\text{VOS}}(t_0, t_f)$.

Similarly, for $Q_3(\theta, t)$, using Equation 2.11

$$\frac{\partial Q_3}{\partial t} = -\frac{1}{t^2}\begin{bmatrix} (x_0 + \rho) + \rho\cos(\omega t) + \rho\omega t\sin(\omega t) + r\sin\theta \\ y_0 + \rho\sin(\omega t) - \rho\omega t\cos(\omega t) + r\cos\theta \end{bmatrix}.$$

Given a value of t, the necessary condition

$$\frac{\partial}{\partial t}Q_3(\theta, t) \cdot \begin{bmatrix} \sin\theta \\ \cos\theta \end{bmatrix} = 0 \tag{3.17}$$

can be solved numerically for θ, and solutions that are in the domain $\theta \in [-\pi, \omega t]$ are considered valid candidate boundary points.

While the sets of all possible boundary points from segments Q_1 and Q_2 define linear segments given by just their endpoints, the set of candidate points (The dotted blue segments in Figure 3-7) found in Q_3 and Q_4 in general cannot be represented analytically in a more compact form. Note that for $Q_{1,2}$, one could first solve independently for $\theta^\star$, while here, the variables cannot be neatly decoupled. Instead, one must discretize the finite interval $[t_0, \min(\pi/\omega, t_f)]$ at some resolution, and at every discrete value of t, points satisfying Equation 3.16 or Equation 3.17 are found. Solutions points from sequential instances in time are matched together to form candidate boundary segments. The resolution of these segments depend on how finely the time interval is discretized. Note that the duration of this interval is upper-bounded by π/ω, the time it takes the obstacle to turn around, thus capping the computational demands of this process.

iii) Points on $Q_5(t)$: This corresponds to lines 16 - 18 in Algorithm 1.

Finally, points on $Q_5(t)$ for any $t \in [t_0, t_f]$ that may be part of $B_{\text{VOS}}(t_0, t_f)$ need to be found. Again, the definitive necessary condition is

$$\hat{n} \cdot \frac{\partial Q_5}{\partial t} = 0, \tag{3.18}$$

where $\hat{n} = \begin{bmatrix} 0 & -1 \end{bmatrix}^T$ (from Equation 3.7) and $Q_5(t)$ is the horizontal line segment joining $Q_1(-\pi, t)$ and $Q_2(\pi, t)$, or joining $Q_3(-\pi, t)$ and $Q_4(\pi, t)$, depending on which set of domains of θ contain $\pm\pi$ at the given t. Since $Q_5(t)$ is a straight, horizontal segment, this condition holds true when the two endpoints satisfy their respective conditions of being normal to $\hat{n}|_P(\theta) = \begin{bmatrix} \sin\theta & \cos\theta \end{bmatrix}^T = \begin{bmatrix} 0 & -1 \end{bmatrix}^T$.

When $\omega t \geq \pi$, $Q_5(t)$ joins $Q_{2,1}(\pm\pi, t)$, and Equation 3.18 is satisfied if $\pm\pi$ are solutions to Equation 3.12 and Equation 3.13. When this is the case, it is easy to show that all the points in $Q_5(t)$ for $t \geq \pi/\omega$ line up with the linear segment between $Q_{2,1}(\theta^\star, t_1)$ and $Q_{2,1}(\theta^\star, t_f)$ described earlier (where $\theta^\star = \pm\pi$). These points do not need to be accounted for a second time.

When $\omega t \leq \pi$, $Q_5(t)$ joins $Q_{4,3}(\pm\pi, t)$, and Equation 3.18 yields

$$y_0 + \rho \sin(\omega t_h) - \rho \omega t_h \cos(\omega t_h) - r = 0 \tag{3.19}$$

This can be numerically solved for t_h. Using $Q_3(-\pi, t_h)$ and $Q_4(\pi, t_h)$, $Q_5(t_h)$ (the black line in Figure 3-6) is then explicitly found and added to the set of candidate boundary points.

See section 3.2.4 for discussion about implications of using Q_5 to form the simplified velocity region in place of the exact boundary.

iv) Combining the Results: (line 19 in Algorithm 1; see Figures 3-6 and 3-7)

As described in Remark 1, considered together with the curves $C_i(t_0)$ and $C_i(t_f)$, the solutions to Equation 3.12, Equation 3.13, Equation 3.16, Equation 3.17 and Equation 3.19 in their specified domains exhaustively describe all points that could lie on the boundary $B_{\mathrm{VOS}_i}(t_0, t_f)$. These pieces of the boundary are found as segments and points, and they need to be sorted into a continuous curve. After all the components are found, the arguments t and θ for each piece are used to identify its neighboring components, thus allowing the segments and points to be correctly sorted into a single curve.

This loop may intersect itself, in which case the segments on the interior are not parts of the actual boundary BVOS. This phenomenon is perfectly consistent, since the conditions that were solved for were simply *necessary*, but not *sufficient*, conditions for points on BVOS. Intersections on this loop of candidate boundary points form corners in the final boundary and are easily detected by checking each segment in the loop in order. It is easy to identify the segments that are on the interior, and these segments are truncated out of the solution. The remainder of the continuous curve is the perimeter BVOS.

3.2.2 Computational Complexity

For each obstacle, Equation 3.12, Equation 3.13, and Equation 3.19 each need to be solved only once, as a function of the single variable θ. Equation 3.16 and Equa-

tion 3.17 need to be repeatedly solved as functions of θ at discretized instances of t, but the range of t is bounded as $t \in [t_0, \min(\pi/w, t_f)]$. For each t, the process outlined in 3.2.3 gives a straight-forward and efficient formula for finding the solutions $P(\theta, t)$. $C_i(t_0)$ and $C_i(t_f)$ are directly found using Equation 3.10 and Equation 3.15 (plus similar equations for $Q_{1,3}$). All of these computations are very light and can easily be done on-line. Calculating and representing the conditions for each obstacle that guarantee collision avoidance is thus a simple process that can be implemented in real-time at each planning iteration. How a planner then uses and checks these constraints is a separate problem.

3.2.3 Geometrical Interpretation

ii) Points on $Q_{3,4}(\theta, t)$: Consider a point (in velocity space) P on the curve Q_3

$$P(t, \theta) = Q_3(t, \theta)$$

for some t and θ such that P is on the boundary B_{VOS}. The solution for P on Q_4 is derived the exact same way after flipping two signs in the x coordinate (See Equation 3.15). By definition Equation 3.2,

$$P(t, \theta) = \ Q_3(t, \theta) = \frac{1}{t} S_3(t, \theta),$$

where $S_3(t, \theta)$ is a point on the circular arc of radius r centered on the bottom corner of the reachability set (See Figure 2-1 and Equation 2.11):

$$S_3(t, \theta) = \begin{bmatrix} x_0 + \rho(1 - \cos(\omega t)) \\ y_0 + \rho \sin(\omega t) \end{bmatrix} + r \begin{bmatrix} \sin(\theta) \\ \cos(\theta) \end{bmatrix}.$$

Thus, the partial derivative $\frac{\partial P(t,\theta)}{\partial t}$ can be re-written as

$$\frac{\partial P(t, \theta)}{\partial t} = \frac{\partial}{\partial t} \left(\frac{1}{t} \begin{bmatrix} x_0 + \rho(1 - \cos(\omega t)) \\ y_0 + \rho \sin(\omega t) \end{bmatrix} \right) + \frac{\partial}{\partial t} \left(\frac{r}{t} \ \hat{n}|_P \right)$$

$$= \frac{\partial \vec{q}}{\partial t} + \frac{\partial}{\partial t}\left(\frac{r}{t}\ \hat{n}|_P\right) = \vec{c} + \vec{b},$$

where $\vec{q}$, $\vec{c}$ and $\vec{b}$ are defined as

$$\vec{q}(t) = \frac{1}{t}\begin{bmatrix} x_0 + \rho(1 - \cos(\omega t)) \\ y_0 + \rho \sin(\omega t) \end{bmatrix}, \tag{3.20}$$

$$\vec{c}(t) = \frac{\partial q}{\partial t} = -\frac{1}{t^2}\left(\begin{bmatrix} x_0 + \rho \\ y_0 \end{bmatrix} + \rho \begin{bmatrix} -\cos(\omega t) \\ \sin(\omega t) \end{bmatrix} + \rho\omega t \begin{bmatrix} \sin(\omega t) \\ \cos(\omega t) \end{bmatrix}\right), \tag{3.21}$$

$$\vec{b}(t,\theta) = \frac{\partial}{\partial t}\left(\frac{1}{t}(r\ \hat{n}|_P)\right) = -\frac{r}{t^2}\ \hat{n}|_P, \quad \|\vec{b}(t)\| = \frac{r}{t^2} \tag{3.22}$$

(see Figures 3-8 and 3-9 for geometric representation of these vectors). Note that the directionality of $\vec{b}$ depends on θ (through $\hat{n}|_P$), whereas the magnitude is purely a function of t. As derived earlier (Equation 3.8), the necessary condition for P to be a boundary point is

$$\frac{\partial P(t,\theta)}{\partial t} \cdot \begin{bmatrix} \sin(\theta) \\ \cos(\theta) \end{bmatrix} = 0,$$

i.e., the vector $\frac{\partial P(t,\theta)}{\partial t}$ must be perpendicular to the normal vector $\hat{n}|_P$, which is parallel to $\vec{b}$:

$$\frac{\partial P(t,\theta)}{\partial t} = \vec{c} + \vec{b} \perp \hat{n}|_P \parallel \vec{b}.$$

Hence, the vectors $\frac{\partial P(t,\theta)}{\partial t}$, $\vec{c}$, $\vec{b}$ form a right triangle with hypotenuse $\vec{c}$ (See Figure 3-8), and an angle α between legs $\vec{c}$ and $\vec{b}$ given by

$$\alpha(t) = \arccos\left(\frac{\|\vec{b}(t)\|}{\|\vec{c}(t)\|}\right). \tag{3.23}$$

Thus, given t, it is computationally trivial to evaluate $\vec{q}$, $\vec{c}$, $\vec{b}$, and α using Equation 3.20 to Equation 3.23. With $\vec{q}$, $\vec{c}$, and α (and the parameter r), it is simple to

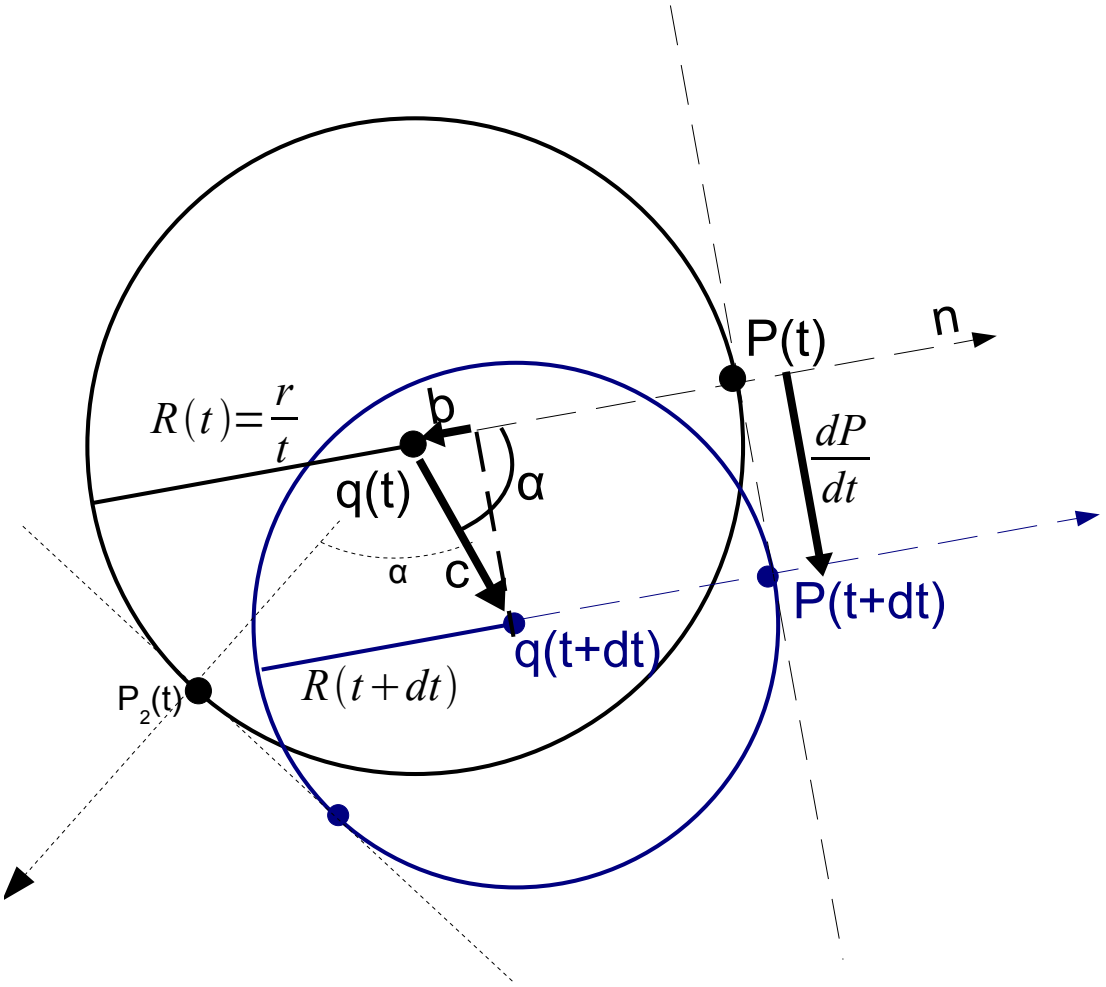

Figure 3-8: How to find candidate points P on $Q_{3,4}$ geometrically. Given t, evaluate q, $\|b\|$, and c; solve for α; then reconstruct P (and second solution P_2) using q and α. See Figure 3-9 to see how q fits into the overall picture.

solve for point P using

$$\begin{aligned} \beta &= \arctan\left(\frac{c_y}{c_x}\right) + \alpha \\ P &= \vec{v} + \frac{r}{t}\begin{bmatrix}\cos(\beta)\\ \sin(\beta)\end{bmatrix}. \end{aligned} \tag{3.24}$$

This yields an analytic process for finding points $P(t,\theta) = Q_3(t,\theta)$ that satisfy the conditions in Equation 3.16 and Equation 3.17, such that they are potentially on the boundary BVOS. As described in Section 3.2.1, the time interval $[t_0, \frac{\pi}{\omega}]$ needs to be

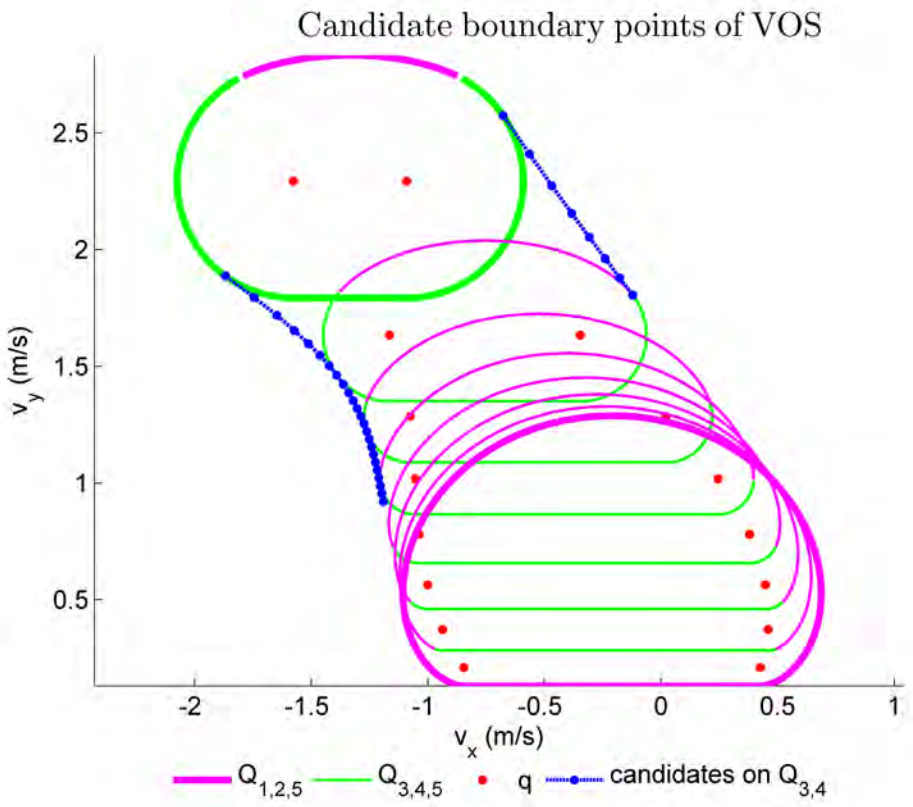

Figure 3-9: Candidate boundary points on $Q_{3,4}$. See Figure 3-10 for context of this detail. 8 instances of SVR(t) are plotted for times between $t_0 = 3$, $t_f = 20$, with the first and last instances thickened. The boundary of each SVR consists of $Q_{1,2}$ in magenta, and $Q_{3,4}$ in green, with Q_3 on the left and Q_4 on the right. The straight horizontal green segment is Q_5. Note that $Q_{3,4}$ are circular arcs centered on q. The candidate points P found are plotted in blue. The discretization of time to find these points P is finer than the discretization for the SVRs shown. $q(t)$ is plotted as a red dot for each SVR. In this particular example, note how there are valid solutions on Q_3 up to roughly $t = 10$ (halfway between t_0 and t_f), and there are valid solutions on Q_4 up to roughly $t = 5$. The process outlined here finds 2 points at each time instance (P and P_2 in Figure 3-8), but in this example, one of the two at every instance lies outside the domain of θ for $Q_{3,4}$ and is discarded (not plotted). These discarded points would have been found on extensions to the circular arcs $Q_{3,4}$ that would be on the interior of each SVR(t), and hence not relevant as candidate boundary points.

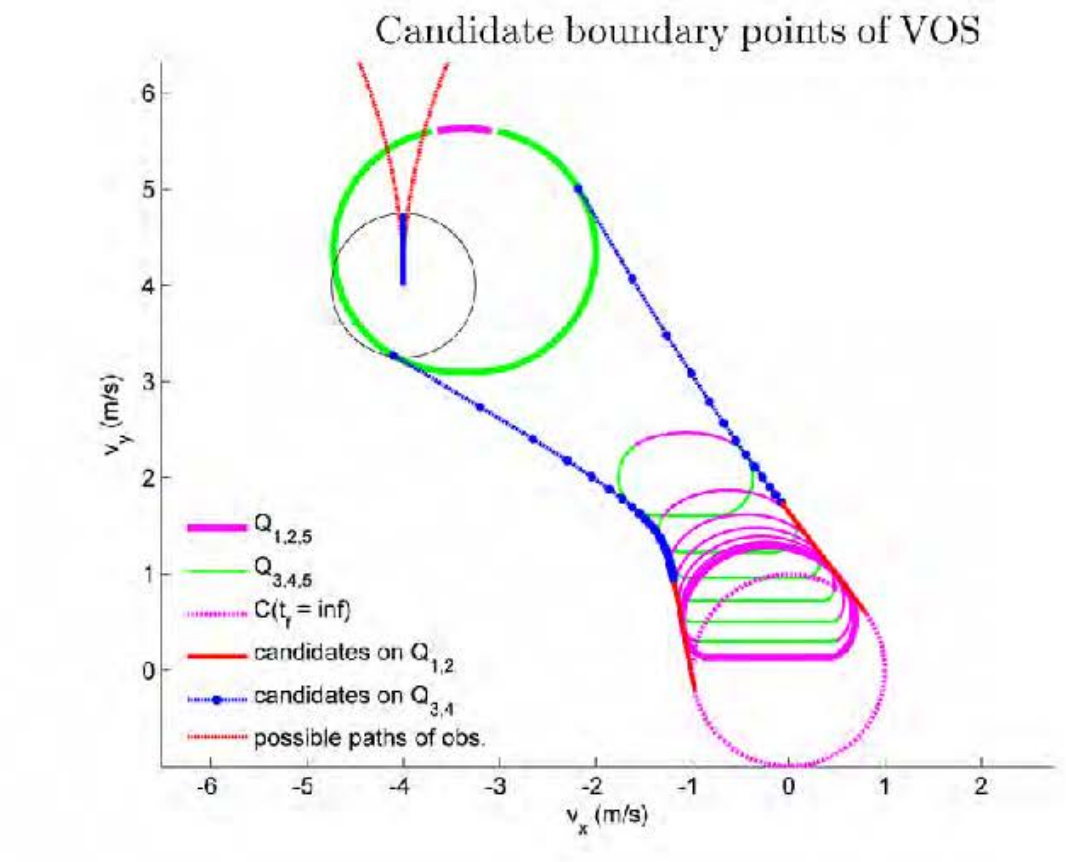

Figure 3-10: Candidate boundary points for $x_0 = -4$, $y_0 = 4$, $\rho = 6.063$, $r = 1.5$, $v = 1$, the same situation as the detail shown in figure 3-9. Note that in this case all candidate points are in fact on the boundary.

discretized, and candidate points $P(t)$ can be found using Equation 3.24 for each discrete value of t.

Note that in Equation 3.23, if $\|\vec{b}(t)\| < \|\vec{c}(t)\|$ there are two values of α and hence distinct two candidate points $P(t)$; if $= \|\vec{b}(t)\| = \|\vec{c}(t)\|$ there is one solution for $P(t)$; and if $\|\vec{b}(t)\| > \|\vec{c}(t)\|$ there are no valid candidates. The corresponding argument θ of each point P obtained through Equation 3.24 still needs to lie within the domain described in Equation 3.14 to be considered a valid candidate point (See Figure 3-9).

3.2.4 Effect of simplifying collision region

If the VOS were defined as exactly the velocities that could contact the reachable set of the obstacle at some time (using the collision region CR(t) instead of the

simplified collision region SCR(t)), any points added from $Q_5(t)$ for some t would not be boundary points of the velocity obstacle set. Instead, these candidate boundary points from Q_5 replace boundary segments that lie in the interior of the simplified VOS, which would take a lot more effort to compute (see Figures 2-2 to 2-4).

To find these candidate points on the exact boundary, first the limits on the domains of $A_{1,2}$ and $B_{1,2}$ would have be computed for $|\theta| > \pi$ (see Section 3.2.1 in [4]). Then, to find possible boundary points within these domains, the computational burden would be very similar to that of finding points on $Q_{1,2,3,4}$; the major hurdles lie in re-deriving all the equations and conditions using Equation 2.3 and Equation 2.4, and, in the implementation itself, of combining points and segments together into a continuous boundary. Tracing the exact curve of the reachable set would imply that θ no longer uniquely defines a point on $C(t)$ for a given t (SCR(t) is convex, whereas CR(t) is not; see Figure 2-2), and additional rules would need to be built into the function that sorts candidate points and segments into an ordering to form a continuous loop.

Note that, especially in cases for which $t_f = \infty$, very rarely (empirically, less than 1% of the time) do any points from Q_5 make it onto the final boundary B_{VOS}. As is the case in Figures 3-5 and 3-6, the candidate points on Q_5 tend to end up within other candidate boundaries formed by points on $Q_{1,2,3,4}$, and are thus excluded from B_{VOS}. When no points from Q_5 are found on the final boundary, none of the points of the original collision region omitted in the simplification could have been on the boundary either, so the VOS found using the simplification is then in fact identical to what would have been found using the exact collision regions, i.e.,

$$\text{VOS}(t_0, t_f) = \bigcup \text{VO}|_{\text{traj}}(t_0, t_f).$$

While these points from Q_5 seldom affect the final VOS, it is nonetheless crucial for the sake of completeness that the algorithm finds these candidate points before discarding them, for otherwise the proof of collision avoidance would be invalidated. The simplification made by using Q_5 is well suited for serving this role as an intermediate

step in the proof.

When the VOS is calculated with a finite time horizon, it is fairly common for the segment $Q_5(t_f)$ to appear on $B_{\text{VOS}}(t_0, t_f)$. In general, these points do not satisfy Equation 3.9, but all points on $C(t_f)$ are also considered candidate boundary points, and $Q_5(t_f)$ is a component of $C_(t_f)$ for any $t_f < \infty$ (see Figure 3-11). When this is the case, a small part of the boundary of the exact collision region lies on the interior of the VOS as defined, so

$$\text{VOS}(t_0, t_f) \supset \bigcup \text{VO}|_{\text{traj}}(t_0, t_f),$$

such that the VOS as defined forms a (very slightly) conservative bound on the single-velocity trajectories that are safe up to time t_f.

3.3 Upper and Lower limits of time window

3.3.1 Lower limit of time window

Just as in the original velocity obstacle formulation [5], as $t_0 \to 0^+$, the velocity obstacle set $\text{VOS}(t_0, t_f)$ extends out to infinity in the direction of the current location of the physical obstacle (reviewed in Section 2.2). This comes from increasingly higher speeds required to collide with the obstacle on decreasing time scales, and beyond a certain point, these velocities are irrelevant since they are beyond the capacities of the host vehicle; these potential collisions as $t \to 0$ are not physically possible. The velocity space constraint $\text{SVR}(t = 0)$ is undefined, since contact between the vehicles at this time depends only on the initial conditions, not any input velocity.

As $t = 0$ cannot be used, a practical approach is to set t_0 at some $\epsilon > 0$ such that $\text{SVR}(t_0 = \epsilon)$ lies beyond input velocities on interest. One method is to set the entire region $\text{SVR}(\epsilon)$ beyond some maximum speed $v_{\max}$ that the host vehicle can achieve (In Figure 3-12, ϵ is chosen such that the red boundary of $\text{SVR}(t_0 = \epsilon)$ is outside of the circle of the host robot's maximum speed).

This is more easily done by considering the dynamic obstacle without turn-rate

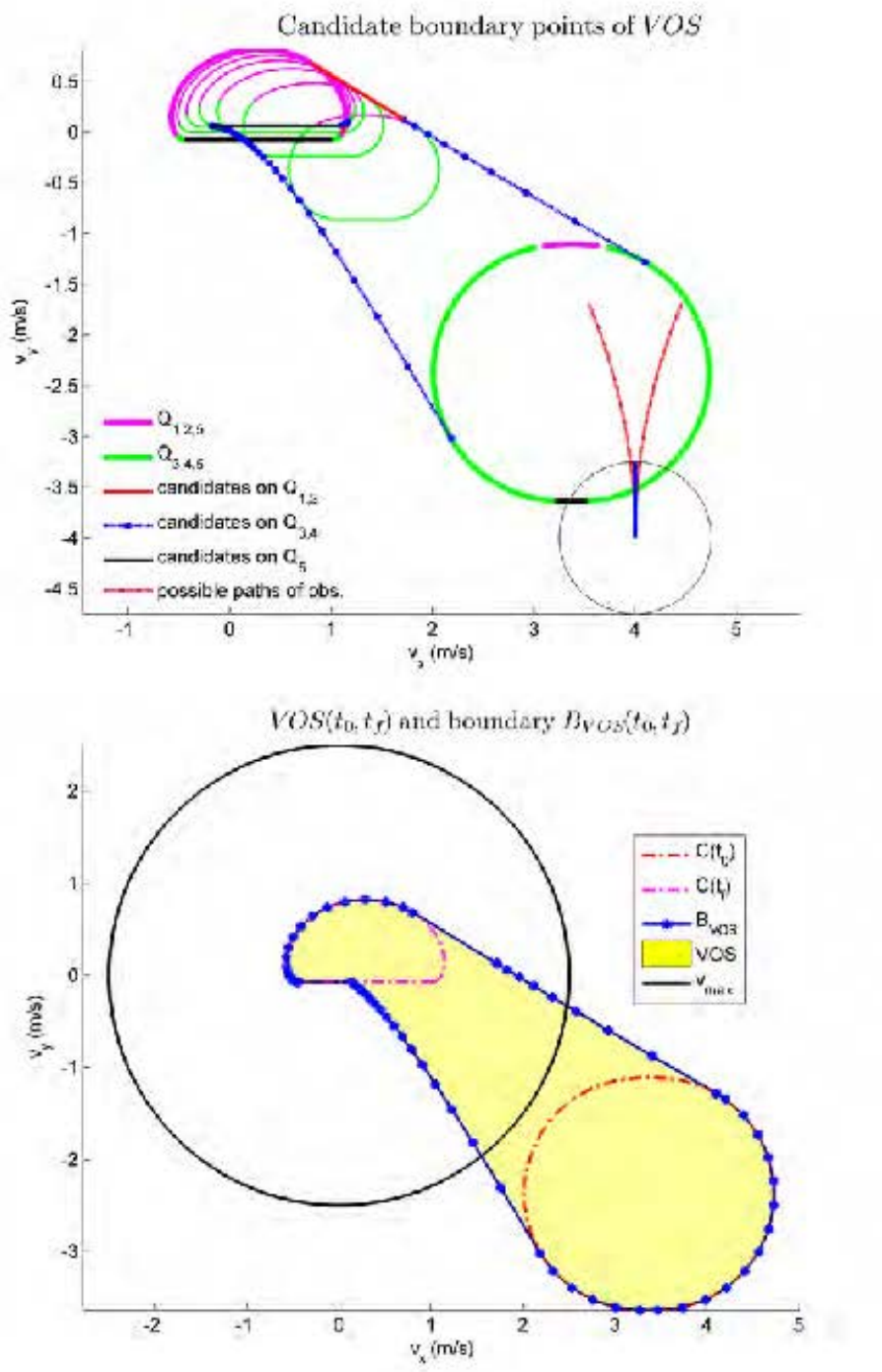

Figure 3-11: Here, $x_0 = 4$, $y_0 = -4$, $\rho = 6.063$, $r = 1.5$, $v = 1$ as in figure 3-6, but $t_f = 14sec$. The simplifying segment $Q_5(t_f)$ is part of $C_(t_f)$, hence the simplification does indeed affect the final boundary. Without the simplification, the less conservative final boundary would include a curved segment that lies on the interior of the straight segment. This curve cannot be solved for using the methods presented in this thesis.

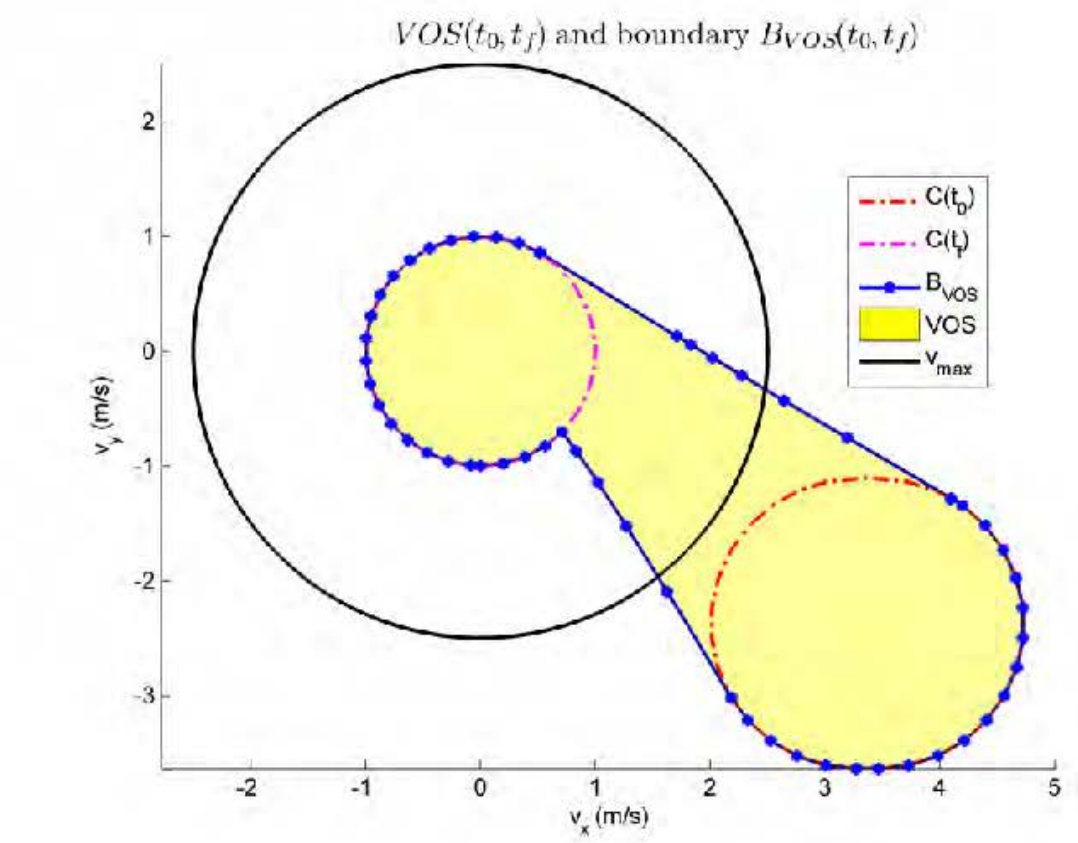

Figure 3-12: $\text{VOS}(t_0, t_f)$ computed for obstacle parameters $x_0 = 4$, $y_0 = -4$, $\rho = 6.063$, $r = 1.5$, $v = 1$. Using the process detailed in Section 3.3.1, $t_0 = 1.188$ was chosen such that $\text{SVR}(t_0)$ lies outside of the set of feasible host vehicle velocities.

constraints and with maximum speed v. The collision region (the reachable set of the obstacle, grown by the collision radius r) associated with a vehicle subject to fewer dynamic constraints contains the collision region of the more limited vehicle, since the reachable set of the latter is a subset of the reachable set of the former. Therefore, if the collision region at time ϵ of a vehicle moving with arbitrary speeds $\leq v$ lies entirely outside of the circle of radius v_{max}, the region $\text{SVR}(\epsilon)$ for a vehicle with a limited turn rate and constant forward speed v that starts at the same location $\begin{bmatrix} x_0 & y_0 \end{bmatrix}^T$ also lies outside of this circle.

The reachable set at any time t of a vehicle with only a maximum speed constraint starting at $\begin{bmatrix} x_0 & y_0 \end{bmatrix}^T$ at $t = 0$ is simply a disk of radius vt centered on $\begin{bmatrix} x_0 & y_0 \end{bmatrix}^T$ (see Figure 3-13). The collision region around this reachable set has radius $vt + r$, so in velocity space, the set of velocities that would enter this collision region at time t is

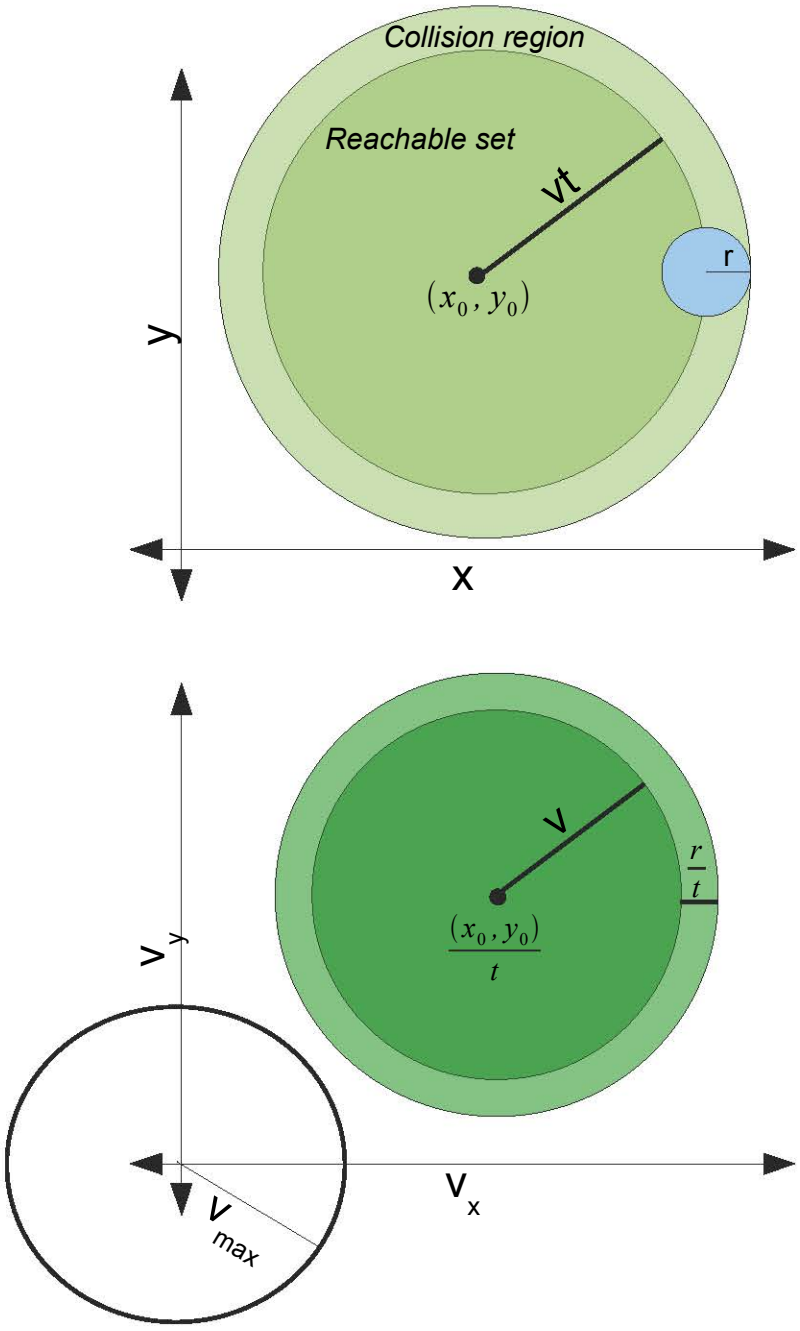

Figure 3-13: Reachable set, collision region, and collision velocities of a moving obstacle with no turn-rate constraints and a maximum speed of v. If this region in velocity space lies outside of the circle of radius v_{max} centered on the origin, then SVR(t) for the dynamically constrained obstacle also lies outside of the v_{max} circle.

a disk of radius $v + \frac{r}{t}$ centered on $\frac{1}{t}\begin{bmatrix} x_0 & y_0 \end{bmatrix}^T$. Thus, the boundary of the collision region in velocity space is given by the parametric curve

$$\frac{1}{t}\left(\begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + (r + vt)\begin{bmatrix} \cos\phi \\ \sin\phi \end{bmatrix}\right)$$

for $\phi \in [0, 2\pi)$. Therefore, to choose $t_0 = \epsilon$ such that this disk is outside of the host vehicle's feasible velocities, solve for ϵ such that

$$\frac{1}{\epsilon}\left| \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + (r + v\epsilon)\begin{bmatrix} \cos\phi \\ \sin\phi \end{bmatrix} \right| \geq v_{\max} \quad \forall\phi \in [0, 2\pi].$$

This yields the formula

$$\epsilon \leq \frac{\sqrt{x_0^2 + y_0^2} - r}{v_{\max} + v}.$$

3.3.2 Upper limit of time window

For an obstacle moving along a known, linear trajectory, the velocity obstacle cone converges to an apex at $\vec{v}_{obs}$ as $t \to \infty$ [5]. For the velocity obstacle set described here, $\text{SVR}(t \to \infty)$ goes to a circular disk of radius v centered on the origin. This can be easily derived by taking the limit of Equation 3.10 (noting that as $t \to \infty$, $C(t) = Q_{1,2}(t, \theta)$ since the domain of $Q_{3,4}$ disappears at $t = \pi/\omega$):

$$\begin{aligned}\lim_{t\to\infty} Q_2(\theta, t) &= \lim_{t\to\infty} \frac{1}{t}\begin{bmatrix} x_0 + \rho(1 - \cos\theta) + (vt - \rho\theta + r)\sin\theta \\ y_0 + \rho\sin\theta + (vt - \rho\theta + r)\cos\theta \end{bmatrix} \\ &= \begin{bmatrix} v\sin\theta \\ v\cos\theta \end{bmatrix}\end{aligned}$$

Intuitively, $\lim_{t\to\infty} \text{SVR}(t)$ is this circle of radius v because, if the host vehicle were to drive along any straight trajectory in any direction with a constant speed less than v, the obstacle could always eventually catch up and collide. This implies that,

to guarantee avoidance of the dynamic obstacle on the infinite horizon without the host robot changing its velocity, the host robot must be able to drive faster than the obstacle.

Choosing a velocity that is not inside of the *infinite horizon velocity obstacle set* $\text{VOS}(t_0, t_f = \infty)$ provides the host vehicle with a single-velocity trajectory that will never enter the collision region of the moving obstacle. The fact that $\text{VOS}(t_0, t_f = \infty)$ is well defined is a very powerful result for infinite horizon safety, which is discussed rigorously in Chapter 4.

Chapter 4

Safety Guarantees

4.1 Introduction

As explained in the beginning of Section 3.1, any velocity outside of $\mathrm{VOS}_i(t_0, t_f)$ for a given obstacle i can be used to generate a single-velocity trajectory that is guaranteed to avoid any dynamically feasible path of that obstacle, for the duration of the time window. For multiple obstacles, the constraints are simply combined; a velocity that is outside of the VOSs of each of the obstacles is guaranteed safe for the duration. The only remaining problem is that, if t_f is set to some finite value τ, there is no guarantee that this safety can be propagated. For any $t' \leq \tau$, the host vehicle would avoid all collisions, but nothing can be said about the continued existence of safe trajectories of duration $\tau - t'$ or longer. This is the major problem encountered by other collision avoidance methods that use finite time horizons τ [9].

However, in this formulation, it is perfectly tractable (Section 3.3.2) and reasonable (because there is no prediction of obstacle behavior involved) to set t_f at ∞, and this allows the use of a slightly modified version of the concept of *invariant safe states* (Section 2.3). As described in [2], these are host vehicle states that

- and can always be propagated back to themselves.
- meet all imposed safety conditions,

In [2], the imposed condition for urban driving is that the vehicle does not contact

other vehicles if the other vehicles maintain their current trajectory; this thesis deals with relaxing such an assumption about the dynamic obstacles. Additional safe states are found as those that can be propagated into some invariant safe state.

4.2 Modified Invariant Safe States

For the scenario described earlier in Section 1.2, once the infinite horizon velocity obstacle sets are found, any control input $\vec{v}^\star$ outside of $\text{VOS}_i(t_0, \infty)$ for all obstacles i can be propagated indefinitely without possibility of collision, simply by definition of the infinite horizon velocity obstacle. Thus, **moving along this trajectory** defined by $\vec{v}^\star$ (the dashed arrow in Figure 4-1) **can be considered an invariant safe state**, where "state" does not imply a specific coordinate or time, as it might in the classic definition. Instead, assuming the host robot starts at the origin at $t = 0$, the "state" $s_{\vec{v}^\star}(t)$ for $\vec{v}^\star$ is defined as

$$s_{\vec{v}^\star}(t) = \begin{bmatrix} [\vec{x}(t) - t\vec{v}^\star] \\ [\vec{v}(t)] \end{bmatrix},$$

where $\vec{x}(t), \vec{v}(t)$ are respectively the host vehicle's position and velocity at time t.

More generally, $s_{\vec{v}^\star}(t_1, t)$ is defined as

$$s_{\vec{v}^\star}(t_1, t) = \begin{bmatrix} [\vec{x}(t) - (\vec{x}(t_1) + (t - t_1)\vec{v}^\star)] \\ [\vec{v}(t)] \end{bmatrix},$$

where $t > t_1$, and t_1 is the time of computation of the VOS.

This redefinition allows a single collision-free state to remain time-invariant for $t > t_1$. Indeed, if the vehicle drives at constant velocity $\vec{v}^\star$, then $\vec{x}(t) = \vec{x}(t_1) + (t - t_1)\vec{v}^\star$, so

$$s_{\vec{v}^\star}(t_1, t) = \begin{bmatrix} \begin{bmatrix} 0 & 0 \end{bmatrix}^T \\ [\vec{v}^\star] \end{bmatrix},$$

$$\frac{ds}{dt} = \begin{bmatrix} \begin{bmatrix} \vec{v}(t) - \vec{v}^{\star} \end{bmatrix} \\ \begin{bmatrix} \frac{d\vec{v}(t)}{dt} \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} 0 & 0 \end{bmatrix}^T \\ \begin{bmatrix} 0 & 0 \end{bmatrix}^T \end{bmatrix}.$$

As stated earlier, being in the invariant safe state $s_{\vec{v}^{\star}}(t) = \begin{bmatrix} [0\ 0] & [\vec{v}^{\star}]^T \end{bmatrix}^T$ simply corresponds to the robot moving with the guaranteed safe velocity $v^{\star}$, and by not changing its velocity, the robot stays in this state. Note that every time the infinite horizon velocity obstacle sets are computed, each velocity that lies outside of all the VOSs defines a new invariant safe state $s_{\vec{v}^{\star}}(t_1, t)$ that the host robot may choose to transition into. Intuitively, these invariant safe states are escape paths (with speed greater than the forward velocity of the obstacle; see Section 3.3.2) that eventually leave all the obstacles behind the host vehicle.

Alternatively, without restructuring the definition of "state," there would be no *invariant* safe states, since the host robot needs to continue moving along the single-velocity trajectory to ensure collision avoidance. Nonetheless, all the vehicle configurations along this trajectory are each safe states, since they can all be propagated into other collision free states by continuing along the trajectory at velocity $\vec{v}^{\star}$. The above modification to the definition of "state" is made in order to avoid the awkwardness of defining safe states without having any invariant safe states to use as a basis.

4.3 Additional Safe States

In addition to the invariant safe states themselves, any vehicle configuration that can safely make a transition into a invariant safe state is also safe; from any of these safe configurations, it is always possible to avoid potential collisions simply by making the transition to an invariant safe state and staying there (i.e., continuing with velocity $\vec{v}^{\star}$). This means that an arbitrary finite trajectory from $t = 0$ to $t = t_1$ (traj_1 in Figure 4-1) terminating with the conditions

$$\vec{x}(t_1) = \vec{x}_0 + \vec{v}^{\star} t_1 \tag{4.1}$$

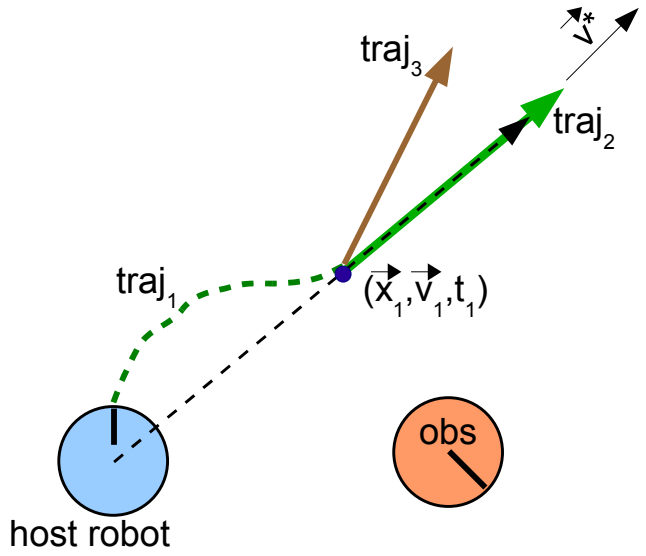

Figure 4-1: Different safe trajectories in physical space, velocity obstacles not shown. Let $\vec{v}^{\star}$ be a velocity outside of the infinite horizon VOS calculated at $t = 0$. The dotted arrow is the single-velocity escape trajectory associated with $\vec{v}^{\star}$, which is guaranteed safe on an infinite time scale. If traj_1 terminates at $(\vec{x}_1, t_1)$ with velocity $\vec{v}^{\star}$, it can be safely continued at least along traj_2. By calculating the shifted velocity obstacle at $(\vec{x}_1, t_1)$, other single velocity trajectories (like traj_3) that also guarantee infinite horizon safety may be found to continue any partial trajectory ending at $(\vec{x}_1, t_1)$.

and

$$\vec{v}(t_1) = \vec{v}^{\star} \tag{4.2}$$

at some value of $t_1 \geq t_0$, without collisions for $t \in [0, t_1]$, is guaranteed to have access to a trajectory that is safe on the interval $t \in [t_1, \infty)$. If t_0 in the VOS computation is set to essentially 0 (using ϵ as described in Section 3.3.1), $\vec{v}^{\star}$ itself trivially produces such a trajectory for $t \in [0, t_1]$, i.e., in Figure 4-1, the dotted black line provides a safe means of arriving to the blue point at t_1. However, various more complicated solutions, like the dotted green line, may be found as well. Depending on the specific scenario and host robot dynamics, various means, including forward simulation and prediction as in [28] and [29], can be used to generate the finite trajectories from $t = 0$ to $t = t_1$ and check for collisions. The velocity obstacle set does not aid in checking the safety of this segment.

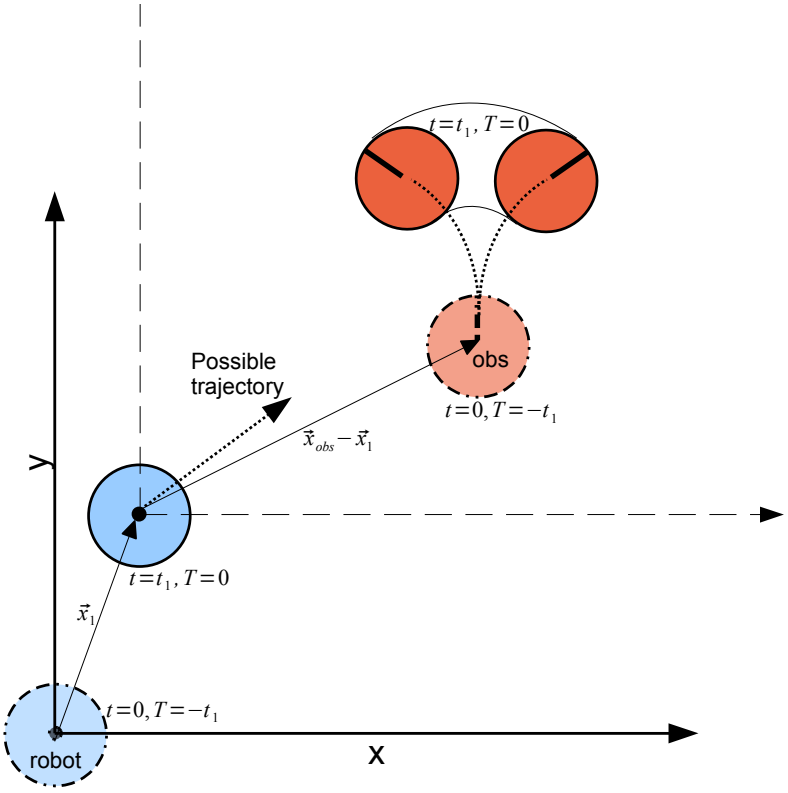

Figure 4-2: For a general coordinate $\vec{x}_1$ that is reached by the host robot at time t_1, a time-shifted VOS can be computed for each obstacle that determines the safety of single-velocity trajectories originating from $\vec{x}_1$ for times $t > t_1$. It is convenient to define a new time-frame $T = t - t_1$ such at $T = 0$, that the host robot is at $\vec{x}_1$ and the reachable set of the obstacle has already grown out of its initial location by t_1 seconds. The time-shifted VOS is computed the same way as detailed in Chapter 3 using these reachable sets for all $T > 0$.

4.4 Time-shifted VOSs

Similarly, but more generally, for any finite trajectory that terminates at time t_1 at an arbitrary position $\vec{x}_1$ with velocity $\vec{v}_1$, the terminal location $\vec{x}_1$ is not in contact with any of the obstacles if it is not in the reachability set $\mathrm{SCR}_i(t_1)$ of any of the obstacles. If this condition is satisfied, then there might be a safe, infinite horizon trajectory on the interval $t \in [t_1, \infty)$ if $\vec{v}_1$ lies outside of every *time shifted VOS* with the host robot moved to $\vec{x}_1$. That is, assuming the robot gets to $\vec{x}_1$ at t_1, one must determine if continuing indefinitely at $\vec{v}_1$ is guaranteed safe for all later times.

To compute the time-shifted VOS of each obstacle, a new time frame is defined

using

$$T = t - t_1,$$

such that T measures the time that has elapsed since the host robot arrived at location $\vec{x}_1$ at time t_1, and the goal is to find velocities leaving $\vec{x}_1$ that are safe for $T > 0$. Each obstacle is translated by $-\vec{x}_1$ such that the coordinates are defined relative to the new host robot location at $T = 0$ (see Figure 4-2). In addition, when computing the reachable sets of the obstacles, any terms T are replaced by $T + t_1$ to account for the motion of the obstacle that has already occurred prior to $T = 0$. When converting these reachability regions defined in physical to velocity space regions, the factor of $\frac{1}{T}$ remains unchanged. Applying these changes to the terms in the equations for $S(t)$ and $Q(t)$ (Equation 2.7, Equation 2.8, Equation 2.11, Equation 2.12, Equation 3.2, and Equation 3.3) yields the boundaries of the time-shifted simplified collision regions $\mathrm{SCR}(T)$.

For example, the expression for $Q_2(\theta, t)$ in Equation 3.10 converts to

$$Q_2(\theta, T, \vec{x}_1, t_1) = \frac{1}{T}\begin{bmatrix} x_0 - x_1 + \rho(1 - \cos\theta) + (v(T + t_1) - \rho\theta + r)\sin\theta \\ y_0 - y_1 + \rho\sin\theta + (v(T + t_1) - \rho\theta + r)\cos\theta \end{bmatrix}, \tag{4.3}$$

where, as before, $\begin{bmatrix} x_0 & y_0 \end{bmatrix}^T$ are the initial coordinates of the obstacle; with domain

$$Q_2(\theta, T, \vec{x}_1, t_1) : T \in [t_0, t_f],\ \theta \in [0, \min(\omega(T + t_1), \pi)]$$

Using the steps given in Section 3.2.1, the necessary condition for boundary points on $Q_2(\theta, T, \vec{x}_1, t_1)$ Equation 3.12 becomes

$$0 = -\frac{1}{T^2}((x_0 - x_1 + \rho)\sin\theta + (y_0 - y_1)\cos\theta + vt_1 + r - \rho\theta. \tag{4.4}$$

Similar modifications can be performed for the segments $Q_{1,3,4,5}(\theta, T, \vec{x}_1, t_1)$, and the necessary conditions for boundary points on each of these segments can be re-derived.

If the velocities obstacle sets are calculated using these modifications, they repre-

sent all the potentially dangerous single-velocity trajectories that originate from $\vec{x}_1$ at t_1, given the current locations of all the obstacles. In an equivalent interpretation of the time-shifted velocity obstacle sets, these are VOSs formed for the current time ($T = 0$) and host vehicle location, but without up to date information of the obstacle locations; the obstacle state is known for $T = -t_1$ but has since changed.

Any velocity that lies outside of the time-shifted VOS can be executed indefinitely without the possibility of collision beyond $t = t_1$. Hence, any finite trajectory terminating at an arbitrary $\vec{x}_1$ at t_1 can be safely extended if the terminal velocity $\vec{v}_1$ of this trajectory lies outside of the shifted VOS of each obstacle. Note that the conditions described earlier in Equation 4.1 and Equation 4.2 are a specific case of this condition; $\vec{v}^\star$ is guaranteed to be one, among possibly many others, velocity that is outside of the VOSs calculated at $(\vec{x}_1, t_1)$. In Figure 4-1, $\vec{v}^\star$ gives the safe trajectory traj_2, and other safe trajectories like traj_3 may be found using the shifted VOSs.

The time-shifted VOS can be used in on-line planning to account for delays; the planner can assign a duration of time for computation, use that as t_1, and set $\vec{x_1} = \vec{v}t_1$ as the expected propagation, where $\vec{v}$ is the current velocity. Assuming the current velocity is maintained during planning, this formulation properly handles on-board computation time and does not compromise guaranteed collision avoidance.

The time-shifted VOS also has potential to be a powerful tool for efficient path planning while maintaining guaranteed safety; finding safe endpoints of general finite trajectories allows for much smoother planning compared to picking only straight trajectories using the various guaranteed safe velocities. This will be discussed in detail in Chapter 5.

Chapter 5

Iterative Planning

5.1 Underlying Concepts

A trajectory that satisfies the velocity obstacle set constraint calculated with $t_f = \infty$ for all the obstacles can be continued indefinitely without collision. However, if the goal location does not lie along the current path, or if the goal location is changed, the robot would never reach the goal without picking a different velocity, and the trajectory needs to be updated. There is no guarantee that it is possible to reach the goal safely.

At any point in time, the planner can use the current state information to recompute the VOS of each obstacle. The safety guarantee is that, if the current trajectory had satisfied the infinite horizon VOS constraints at some previous time, the current velocity will still lie outside of all the updated VOSs. In addition, if there are other velocities that satisfy this condition, there may be other **feasible** trajectories that are guaranteed safe as well. These may be single-velocity trajectories using a valid $\vec{v}^{\star}$, or more complex finite paths whose terminal conditions satisfy either Equation 4.1 and Equation 4.2 or the more general requirements given by the time-shifted VOSs (Section 4.4). Here, feasibility refers to scenario-dependent constraints on the host robot's dynamics; the robot may not be able to immediately change its speed and heading to match the desired new safe velocity (see Section 5.2.2 for a detailed discussion). If a significant amount of time or displacement is required to make the transition, either

a finite path that eventually merges onto the new trajectory can be computed and checked for safety (Section 4.3), or the time-shifted VOS can be used.

Typically, after the VOSs are updated, new safe trajectory will be available, in addition to the guaranteed safety of the current trajectory. Clearly, for any safe velocity pointed in the direction of the goal, if the distance to the goal divided by the speed is less than the time horizon of the VOSs, the host robot will safely reach the way-point. If there are no such velocities, a safe velocity can be chosen subject to various heuristics. For example, to choose the trajectory that makes the closest approach to the desired way-point at some future time, the velocity whose direction is most closely aligned with the goal should be chosen. Alternatively, candidate safe velocities can be propagated for the duration of time until the next scheduled re-plan, and a best velocity can be chosen based on the resulting robot configuration. For example, the planner could greedily select the propagated location closest to the goal, or it could also incorporate a measure of how sharply the robot would need to turn to travel from the propagated location to the desired goal. Other selection metrics could include measures of control effort.

At any later time, it is always possible to recompute the VOSs and attempt to improve the trajectory. Re-planning can occur on regular intervals, or it can be triggered by how well the current plan approaches the goal. A better trajectory may not exist, and there is no guarantee that the goal can ever be reached, but at least the current plan can be executed safely and extended indefinitely (if the VOSs were computed for the infinite time horizon). Therefore, after solving for the velocity obstacle sets given the initial state of the environment, if there exists at least one feasible velocity $\vec{v}^{\star}$ outside of all the infinite horizon VOSs, an iterative planner is guaranteed to never encounter collisions. **Note that the safety guarantee does not depend in any way on the iterative updates, which only help the robot seek the way-point.** Other receding horizon methods ([2, 9]) have not been able to provide this guarantee for unpredictable obstacles.

This approach is not *complete*; velocity obstacle sets only return single-velocity escape trajectories. There may exist a curved path that extracts the host robot from

a "surrounded" situation or one that brings it to the goal, but the method presented in this thesis will not find it. However, this method does guarantee that it will never put the robot in such a surrounded situation if that is not the initial condition.

5.2 Outline of Planners Used in the Simulations

An iterative planner based on the concepts discussed in Section 5.1 was implemented in simulation for scenarios with various conditions to demonstrate the use of velocity obstacle sets in avoiding unpredictable, dynamic obstacles. Multiple obstacles are seeded throughout then environment and propagated in accordance with the assumed dynamical model. The host vehicle is given a series of way-points it attempts to reach, if it can do so while avoiding all the obstacles. The planner is provided perfect knowledge of the obstacles' model parameters (size, speed, and turn radius) and current locations and headings such that it can compute the velocity obstacle sets, but it is given no additional information. On regular intervals, the planner computes the VOSs of each obstacle and uses a greedy heuristic to choose and immediately implement a safe, dynamically feasible velocity that brings it towards the next way-point. The key result of this thesis is that, **by computing the infinite horizon velocity obstacles, the host robot can successfully avoid the obstacles ad infinitum while attempting the navigate to the way-points**. Note that just reaching the goal is not considered safe; the robot must maintain an open path forward from there. This allows the simulation to run indefinitely without the robot ever colliding.

The infinite horizon planner is tested without imposing constraints on the host vehicle, and then tested again with turn-rate restrictions on the host vehicle, both cases agreeing with the analytically proven safety guarantee. Then, to examine the trade-off between long-term safety and short-term goal-seeking and to demonstrate the capabilities of the VOS formulation as a reactive planner, finite-horizon VOSs are used in place of the infinite horizon VOSs, and the simulations are re-run using various horizon lengths for both cases where the host robot has unconstrained or

constrained dynamics. Finally, in certain applications, the traversable environment may have enclosing boundaries (walls, or limits of explored terrain), in which case any non-zero velocity would lead to a collision on the infinite horizon. Various length finite-horizon planners with no long-term safety guarantees are tested in simulation for this case and the empirical results are compared.

5.2.1 Implementation Details of the Simulator

This section describes in detail the implementation of the iterative planning algorithms in outlined in Section 5.2.

Basic Parameters In these simulations, six obstacles move with speed $1^{\mathrm{m}}/_{\mathrm{s}}$ and turn at a maximum rate of $^{\pi}/_{5}{}^{\mathrm{rad}}/_{\mathrm{s}}$, yielding an equivalent minimum turning radius of $\rho \approx 1.59$ m. The host robot has a maximum speed of $v_{\max} = 2.5^{\mathrm{m}}/_{\mathrm{s}}$. In the simulations for which the host robot has a limited turn rate, it may adjust its heading immediately by up to $\Delta\theta = {}^{\pi}/_{3}$radians at each planning update. The dynamics are propagated linearly at intervals of $\delta t = 0.1$s while the planner makes updates every $\Delta t = 1$s, so the limited instantaneous heading adjustment translates into an equivalent turn rate of between ${}^{\pi}/_{3}$ and ${}^{10\pi}/_{3}\ {}^{\mathrm{rad}}/_{\mathrm{s}}$ (see Section 5.2.2 for a detailed discussion on how this is obtained). The host robot and the obstacles are all disks of radius 0.5m, yielding a collision radius $r = 1$m. Each simulation runs for a total duration of 800s of virtual time.

Obstacle behavior To represent all feasible obstacle behavior, obstacle turn rates are uniformly sampled from their full range of $[-\omega, \omega]$ every one to two seconds. If new turn-rates were sampled too often, the trajectories would become noisy – but mostly straight – lines, which would not present a challenging collision-avoidance scenario. To keep them clustered in a local area and to sufficiently test the limiting cases, the obstacles are made to turn at maximum rate whenever they are outside of a pre-defined $12^2\mathrm{m}^2$ box (the dashed lines in Figures 5-2-5-10) in at the center of the environment. For the simulations in which the space is enclosed by walls, the

walls form a 19.4^2m^2 centered on the origin (in the figures, the walls are the limits of the viewing window), such that given the above described obstacle behavior, the walls mark the edge of the obstacles' reachability. Collisions between obstacles do not affect their motion so as not to violate the assumed dynamics.

Goal-seeking and collision avoidance for host robot The host robot is repeatedly given the way-points $(0, -6)$, $(6, 0)$, $(0, 6)$, and $(-6, 0)$ in sequence, such that it is continuously led in a counter-clockwise circuit through the region occupied by the moving obstacles. Whenever the host robot successfully gets within 0.1m of the current way-point, the next way-point is given to the robot. At each planning iteration occurring at intervals of $\Delta T = 1\text{s}$, the VOS of time horizon τ for each obstacle is computed according to Algorithm 1. The velocities that lie outside of each $VOS_i(\epsilon, \tau)$ yield single-velocity trajectories that are guaranteed to be collision-free with respect to the obstacles for the desired time horizon. If walls are defined, the velocities are again filtered such that only the those do not reach the walls by time $t = \tau$ are considered safe. Of these safe trajectories, the dynamically feasible ones are those that are less than $v_{\max}$ in magnitude and, if the host robot is described with a limited turn-rate, within $\Delta\theta$ of the previous velocity. Note that the dynamic constraints of the host robot and the collision avoidance conditions are completely decoupled; the host robot's dynamics need not be known to compute the VOSs, so this collision avoidance algorithm can be used modularly in various scenarios.

Dynamically feasible single-velocity trajectories that are guaranteed collision-free up to time τ are ranked using a simple heuristic: each candidate safe trajectory is propagated for a duration of Δt, and the velocity that brings the robot the closest to the current way-point is chosen. This greedy, one-step look-ahead approach is not the time-optimal solution, but in most cases it is a simple approximation that effectively brings the robot quickly to the goal. As one would intuitively expect, the shortcomings of this heuristic are most apparent when a sharp turn is required to reach the way-point. In this thesis, the main focus of the work is about defining the constraints that guarantee collision avoidance, so improved formulations of more

effective path planners are left for future work. Such planners would greatly benefit from allowing general motion primitives that terminate in provably safe single-velocity trajectories (see Sections 4.3 and 4.4).

As a matter of implementation, candidate safe velocities are found as the boundary points of each VOS, as well as the intersection of these boundaries with any other constraints (max host speed, host turn rate limits, collision constraints relative to wall). These velocities are sorted by the metric described above, and candidates are then sequentially checked to see that the velocity indeed satisfies all the constraints (i.e., a boundary point of one VOS does not lie on the interior of a different VOS). When a valid velocity is found, it is randomly perturbed by small amounts until a nearly velocity is found that is not right on the boundary and falls on the correct side of it, such that potential contact with obstacles is completely avoided, whereas a boundary point may allow the host vehicle to just graze the obstacle.

In the case that no dynamically feasible velocities are guaranteed safe for the desired horizon τ, the planner implemented here simply selects the dynamically feasible velocity that best satisfies this goal-seeking heuristic, without attempting to quantify the level of risk for each unsafe trajectory and compute a trade-off with the expected time-to-goal. Note that this situation is proven to never arise if $\tau = \infty$. The authors of Ref. [15] explore in detail how this can be done for velocity obstacles of objects moving along known trajectories. Their method of computing an expected time-to-collision could be applied to regions within the velocity obstacle set and used accordingly, but such an implementation has been omitted here, as the primary focus of this work is guaranteed safety.

Differences for baseline simulation The first set of simulation results presented in Section 5.3 focuses on the baseline characteristics of using infinite horizon VOSs to guarantee collision avoidance with no dynamic constraints nor enclosing walls, and the simulation is implemented with some minor differences in the details described above. There are only four instead of six obstacles, such that the ensuing behavior is easier to understand. Instead of being given a fixed sequence of way-points, the

way-points are generated randomly, with a biased to be close to the current location of the obstacles in order to create interesting collision avoidance scenarios. The best safe velocity is selected using a different heuristic: the velocity most closely aligned with the direction of the goal is chosen such that the resulting trajectory is the one that gets the closest to the goal. If multiple trajectories point directly to the goal, the one with speed closest to $1.5^{\mathrm{m}}/_{\mathrm{s}}$ is chosen such that the host robot is not always operating at its dynamical limits.

5.2.2 Nonlinear motion of the host robot

The core concept behind finding velocity obstacle sets for collision avoidance is that these boundaries in velocity space separate the safe single-velocity trajectories from the potentially dangerous ones. However, there is no simple modification to adapt it to immediately determine the safety of general, nonlinear motion of the host robot. Instead, a nonlinear trajectory would need to be decomposed into general segments and straight, single-velocity components. Safety with respect to the unpredictable dynamic obstacle is determined by checking that, at all times, the robot position lies outside of the simplified collision region $\mathrm{SCR}(t)$ (Section 2.1) of each moving obstacle. For any segment of a general trajectory, this check of physical space constraints can be done in a straight-forward manner by systematically sampling times in the window of interest and computing the collisions regions. For single-velocity segments defined for some interval $[t_1, t_2]$ (where t_2 may be set to ∞), this check may be simpler to perform by computing $\mathrm{VOS}(t_1, t_2)$. Note that if $t_1 \neq 0$, the VOS needs to be calculated as a time-shifted VOS (Section 4.4).

Therefore, to properly handle the non-instantaneous turning of the host robot, the motion should be broken down as a finite curved segment up to some time t_1 terminating at some known location $\vec{x}_1$, followed by a single-velocity trajectory out to infinity. The safety up to t_1 needs to be checked in physical space, and ensuing safety out to $t_f = \infty$ is guaranteed if the final velocity is outside of the shifted VOS found using t_1 and $\vec{x}_1$. This would allow the planner to consider much more general paths, if there were a more sophisticated algorithm in place to generate and rank such

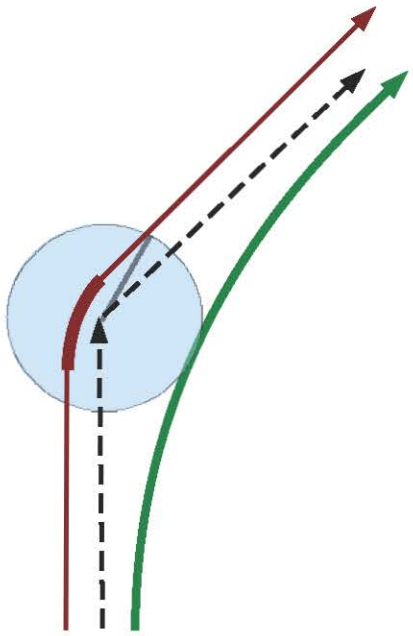

Figure 5-1: Nonlinear piece-wise motion of the host robot. If the robot has unlimited turn rate, the motion can be accurately represented by the dotted black line. If not, the velocity obstacle set calculations do not directly aid in determining the safety along any curved segments, which need to be checked in physical space over whatever times they are defined to be technically correct. Alternatively, in discrete simulation time, the red trajectory is equivalent to the dotted black trajectory, such that curvature constraints of the red curve can be translated into angular constraints at the corner in the linear approximation. The de-facto turn-rate limit of the robot (the green curve) represented by this linear approximation depends on the re-plan frequency and is less sharp.

trajectories.

The added complexity of this formal approach can be bypassed, however, if the exact nonlinear trajectory of turning for a duration of δt and then proceeding straight can be closely approximated by an instantaneous change of direction followed by single-velocity motion (See Figure 5-1, equivalent exact trajectory in red, instantaneous change in dotted black). In a discrete-time simulation with time step δt, they are indeed identical if the instantaneous change in heading $\Delta\theta$ in the piece-wise linear trajectory is less than the actual turn rate ω_{h} of the host robot multiplied by δt. Hence, as described in Section 5.2.1, limiting the choice of new velocities at each re-plan to those within $\Delta\theta \leq \omega_{\mathrm{h}}\delta t$ equivalently represents a limited turn rate of the

host robot that is less than $\omega_{\max} = {}^{\Delta\theta}/_{\delta t}\ {}^{\text{rad}}/_{\text{s}}$, if the actual motion is structured as quick turns of duration δt followed by straight motion.

Alternatively, without being technically identical, restricting the instantaneous change of heading to be at most $\Delta\theta$ can be treated as an equivalent maximum turn rate of $\omega_{\max} = {}^{\Delta\theta}/_{\Delta t}\ {}^{\text{rad}}/_{\text{s}}$ of the host vehicle, where Δt is the time between successive re-plans. That is, given that the host vehicle can only change its velocity one re-plan at a time, the overall rate of angular change of the trajectory over the course of several re-plans (green curve in Figure 5-1) is limited by the product $\Delta t\ \Delta\theta$. However, it is not technically accurate to say that if the host robot has dynamics limited by $\omega_{\text{h}} \leq {}^{\Delta\theta}/_{\Delta t}$ it would be able to follow the piece-wise linear trajectory, which is implemented in the simulator, down to discrete-time accuracy. Instead, a vehicle with such dynamics would be able to use the piece-wise linear trajectory as a reference trajectory, where the actual path traversed would consist of curved segments merging onto the reference trajectory (see Section 4.3) after Δt seconds. The safety along the curved segments would need to be checked by other means, or simply assumed.

For the purposes in this thesis, the distinction between the two interpretations is not very important. For smoother path planning that is capable of considering more general escape trajectories than just the single-velocity ones, it will be worthwhile to have an implementation capable of checking curved segments followed by infinite horizon guarantees provided by time-shifted VOSs. This would be the best way to properly handle realistic turn-rate constraints of the host vehicle, but fully implementing such a planner still requires much additional work. Until then, capturing the qualitative effects of non-instantaneous turning by imposing some maximum $\Delta\theta$ is sufficient for commenting on how dynamic limitations affect the safety and the navigational capabilities of the planner under different conditions.

5.3 Baseline Infinite Horizon Planner

First, the simulation is run using infinite horizon VOSs with no dynamic constraints nor enclosing walls. The safety guarantee is manifest as the fact that, whenever a

re-plan occurs, the current velocity is always among the safe velocities found using the updated VOSs, since the current single-velocity trajectory is guaranteed to remain obstacle-free for all time. Indeed, this simulation runs for the full duration of 1000s of virtual time without the robot colliding with any obstacles, and select screen-shots are shown in the following figures to illustrate the resulting collision avoidance behavior.

5.3.1 Interpretation of Figures

In the ensuing figures, the physical space is shown on the left. The host vehicle is in blue, the obstacles are in magenta, and the way-point is the red star. The vehicles leave a trail of their most recent positions. The trail of the host vehicle is reset when it reaches a way-point and is assigned a new goal. Whenever the obstacles cross the dashed boundary, they are forced to turn around (explained in Section 5.2.1).

On the right is the velocity space. The infinite horizon VOS of each of the four obstacles is shaded in. The maximum velocity $v_{\max}$ of the robot is represented by the black circle. The ideal velocity (directly aligned with the way-point, magnitude equal to preferred speed) is the green vector; the best new safe velocity is the blue vector; and the velocity before the re-plan is the dashed black vector. The safety guarantee is that, until the host vehicle begins traveling along a different velocity, the old velocity will still lie outside of all the VOSs when the obstacle states are updated. Just for directional reference, the nearest obstacles are sketched in dashed outlines, though they cannot be truly represented in velocity space.

5.3.2 Some Sample Snap-shots

Visualizations from various instants in time are presented in Figures 5-2 to 5-10. The planner makes a scheduled update every $\Delta t = 1$s, but the snap-shots may also fall on non-integer times since an update is immediately computed whenever the host robot reaches a way-point.

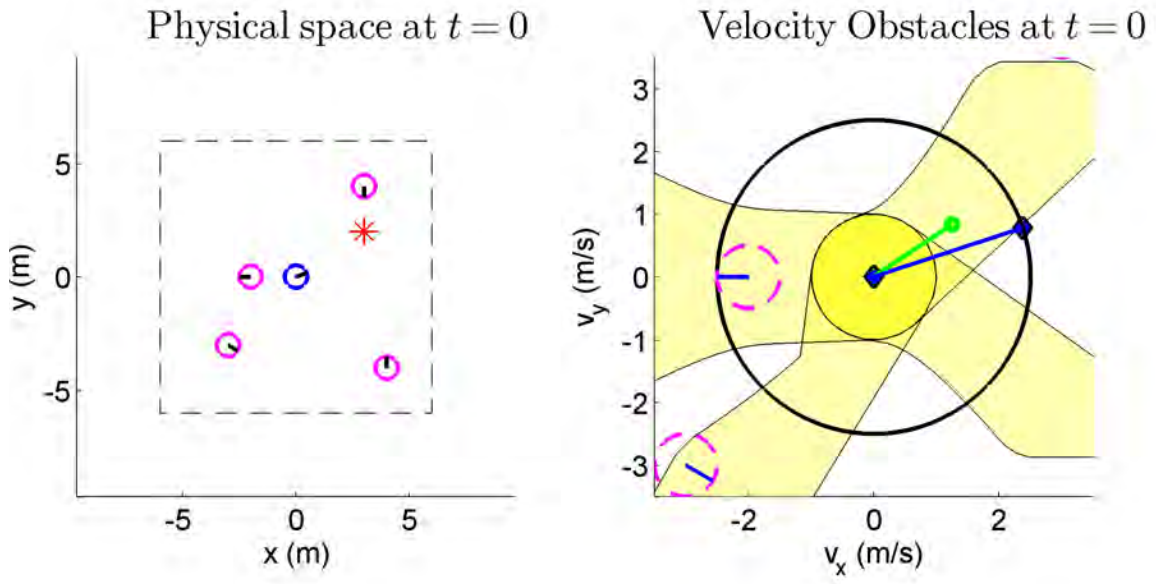

Figure 5-2: Initial condition of baseline simulation. Four obstacles are scattered in the environment, and the host vehicle is given a random way-point. The velocity obstacles show which velocities could potentially result in collisions if propagated indefinitely. Since there is no safe velocity directed straight towards the goal, the planner chooses the most-closely aligned safe velocity. Re-plans in the future may adjust this trajectory to intersect the desired way-point. Notice that the closest obstacle blocks off the largest section of velocity space.

5.4 Performance under Various Conditions

This section will examine the changes in avoidance behavior and the trade-offs between long-term safety guarantees and short-term effectiveness of path planning as the time-horizon used in the planner is varied under different scenarios. In the first batch of test scenarios, the environment is not enclosed by walls, and the host robot can either turn arbitrarily quickly, or it is held to a limited turn-rate. For each case of host robot dynamics, multiple simulations are run using a spectrum of finite time-horizons for the VOSs in addition to the infinite horizon formulation. The minimum finite time-horizon tested is $\Delta t = 1$s, for only avoiding collisions that would occur on a horizon shorter than the time until the next plan would obviously do a poor job at avoiding collisions. As the finite time horizons become large enough, there is no discernible difference between the finite and infinite horizon cases.

A second batch of simulations for which the space is enclosed on all four sides by

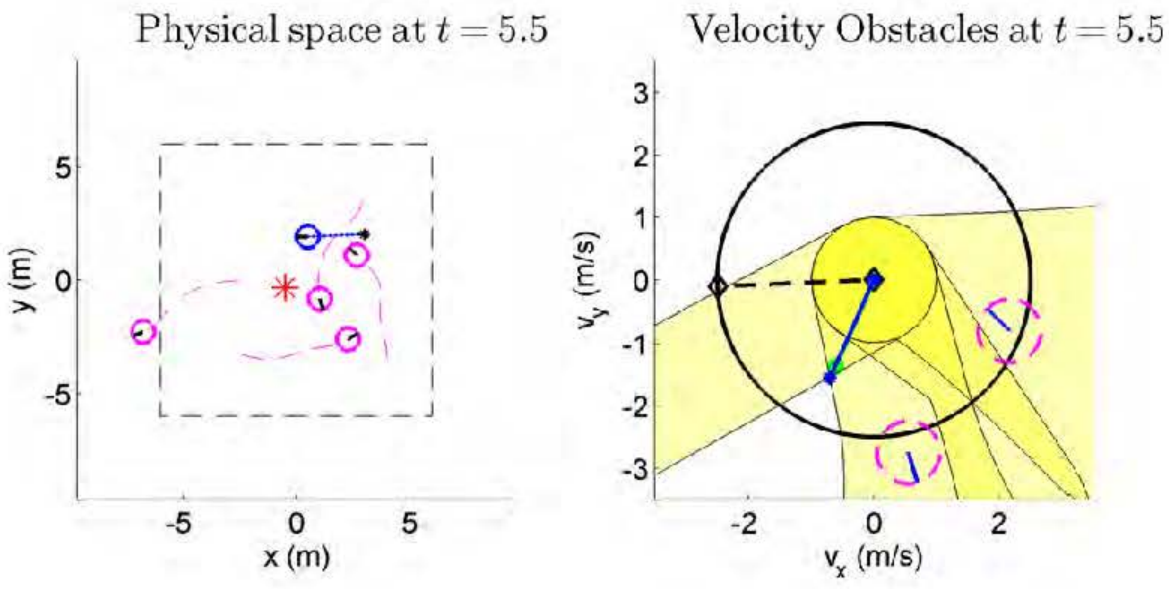

Figure 5-3: At $t = 5.5$s, a velocity pointed directly towards the goal is guaranteed safe, and this new velocity is selected.

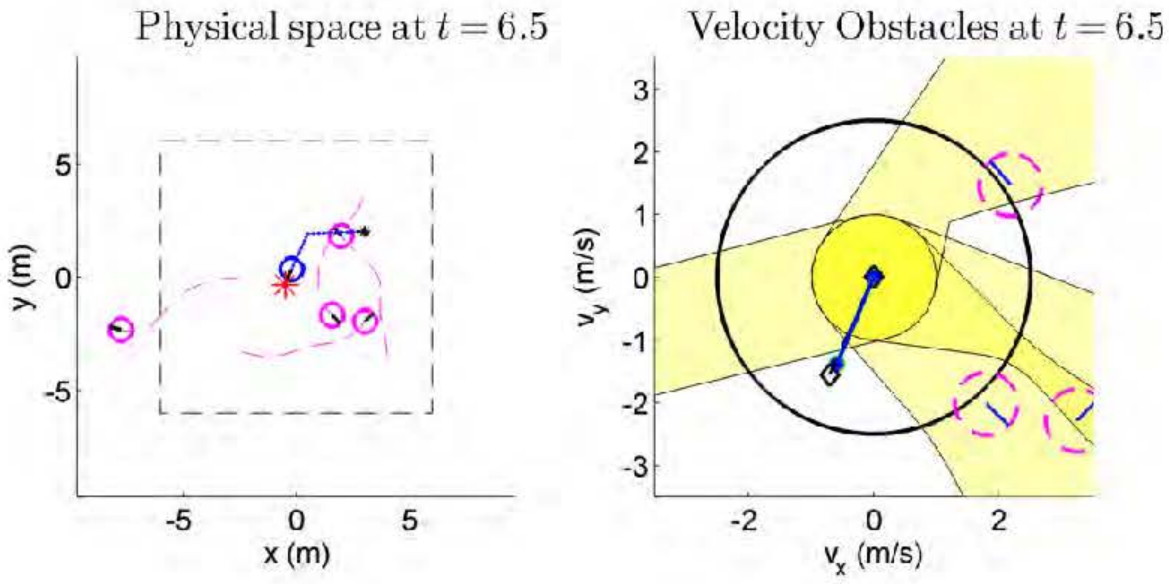

Figure 5-4: By the next planning iteration, the slightly slower ideal velocity had become become safe, and the host robot adjusts accordingly. This change comes from the two left-most obstacles having each turned away from the robot's intended path while it had been dynamically feasible for them to turn in the other direction.

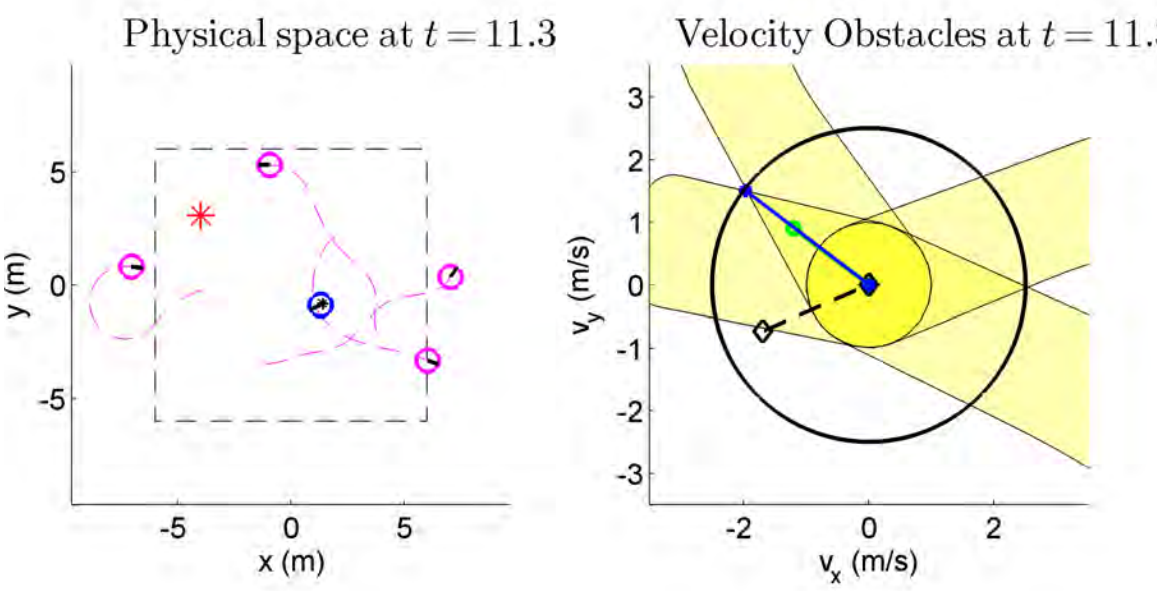

Figure 5-5: At $t = 11.3$s, a velocity is found that squeezes between the two velocity obstacles sets towards the left.

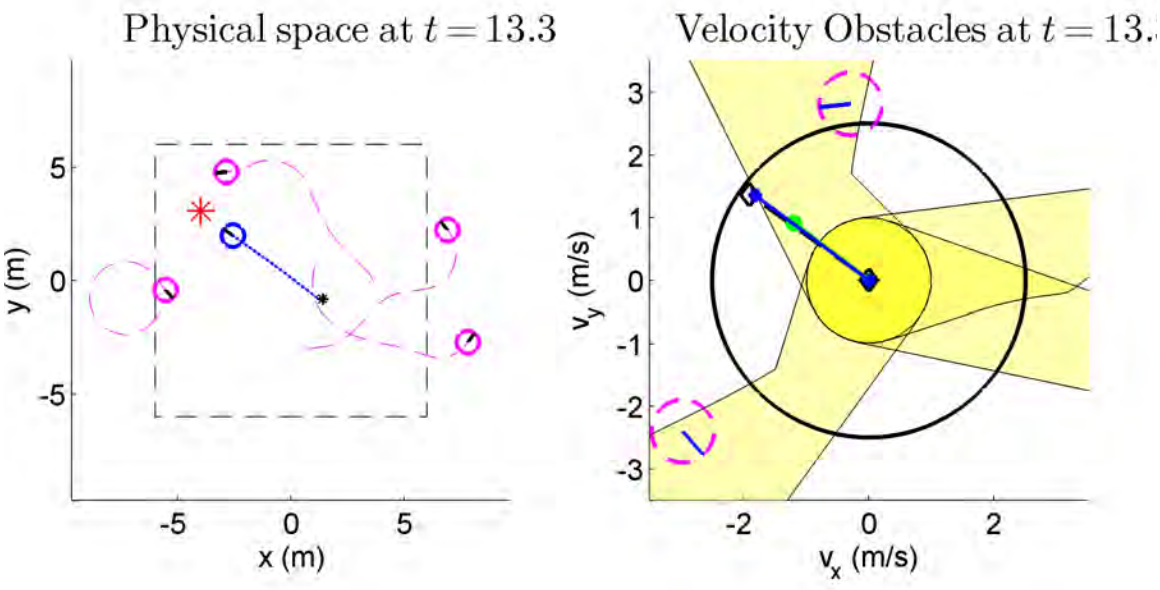

Figure 5-6: Two seconds later, the updated VOSs show that this direct path is indeed still safe, and the actual trajectory of the left-most obstacle has allowed wider margins on the host robot's left side while the upper obstacle continues to threaten collision if the host robot were to turn slightly to the right.

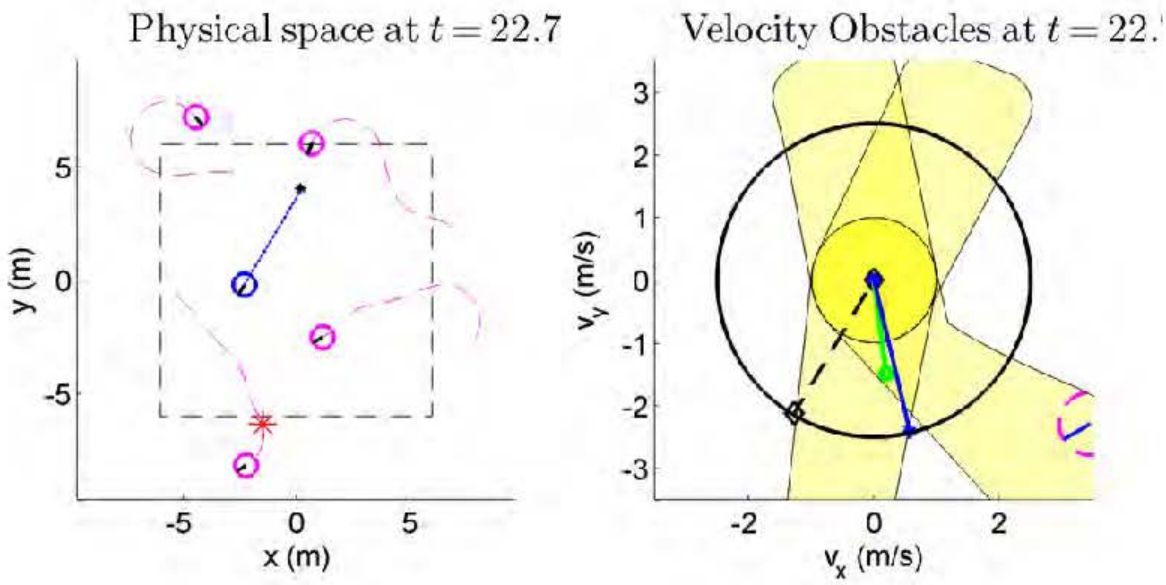

Figure 5-7: In the few planning iterations before $t = 22.7$s, the best safe trajectory that the planner had selected would have brought the robot too far the left of the goal. At $t = 22.7$s, the updated VOSs show that there is now a safe trajectory nearly lined up with the goal that jointly avoids all possible trajectories of both lower obstacles.

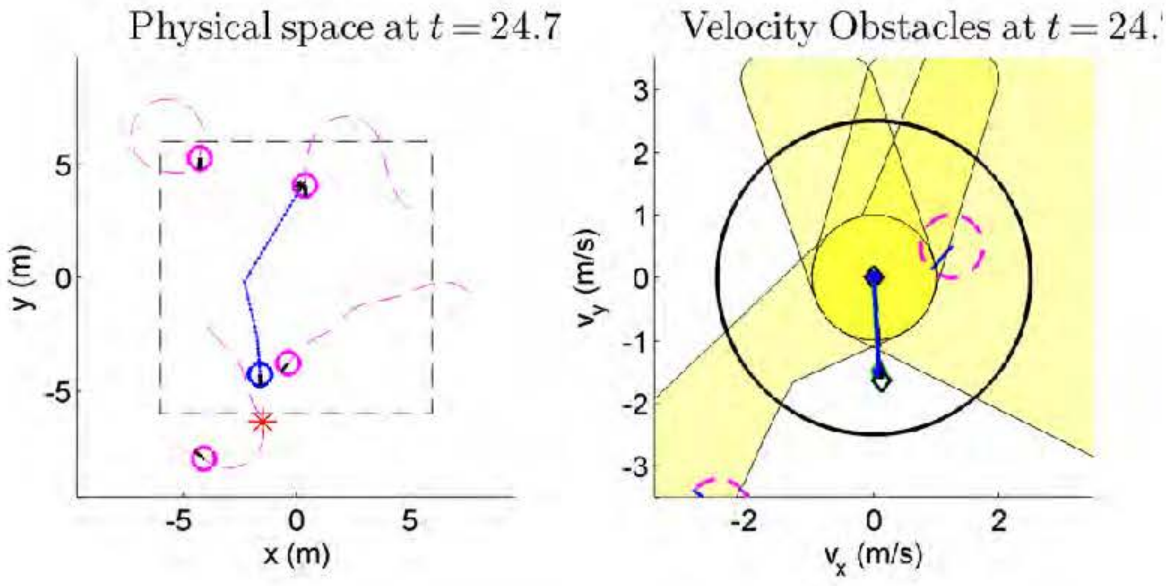

Figure 5-8: In the one re-plan between $t = 22.7$s and $t = 24.7$s the robot was able to make one more minor adjustment in its heading (as seen by the slight deflection of the blue trajectory), lining up directly with the goal.

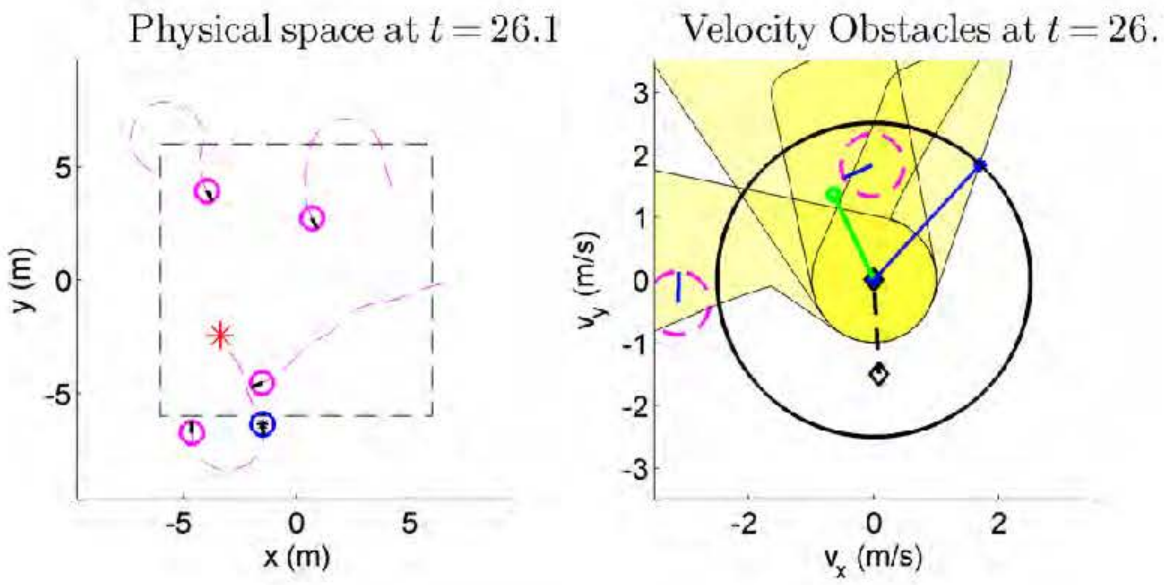

Figure 5-9: At $t = 26.1$s, the new way-point is blocked off by the nearest obstacle, and host robot must drive around it.

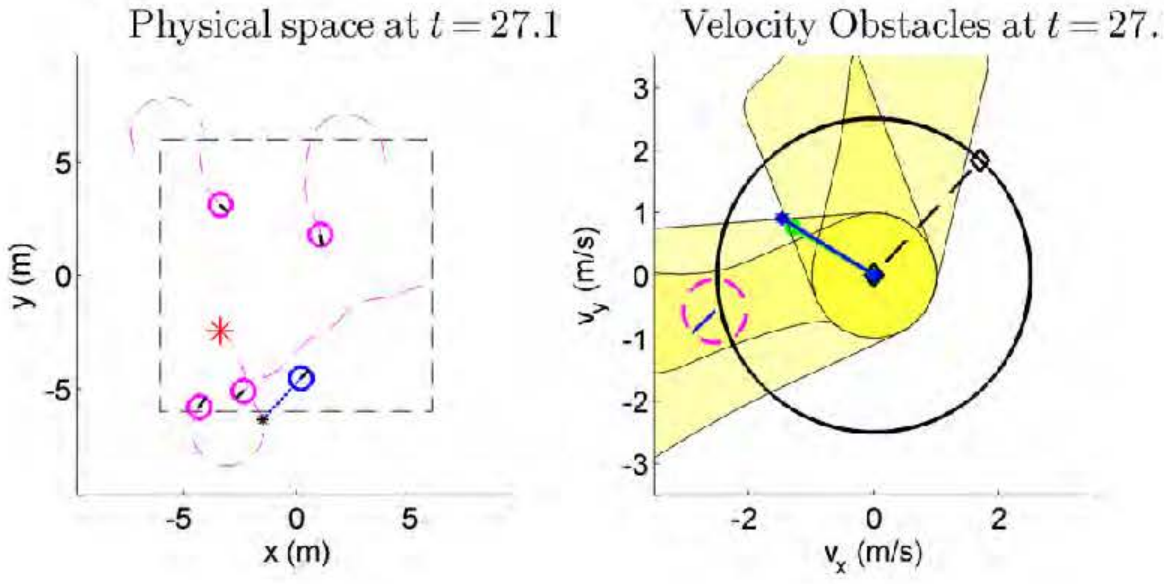

Figure 5-10: At the next update, the planner discovers a clear trajectory towards to goal. Note that, with limited turn dynamics, this tight turn may not be possible, and the planner would then continue along the previous velocity as the next best option.

walls is then analyzed. In these scenarios, it is not possible to form infinite horizon velocity obstacle sets, since any single velocity trajectory would eventually collide with a wall, so various finite time horizons are tested and compared for both constrained and unconstrained host robot dynamics. Note that guaranteed collision avoidance requires infinite horizon VOS calculations and a non-enclosed environment.

5.4.1 Measurements Taken

Measures of Safety To evaluate the safety of the planner in each condition, the number of collisions encountered in the span of 1000 seconds of simulation is recorded. These collisions are either due to planner errors or VOS numerical errors.

- **Planner errors** These occur when there are no velocities guaranteed safe for the desired time horizon τ, i.e., the velocity obstacle sets cover all of the dynamically feasible velocities of the host robot. When this occurs, the planner selects an unsafe velocity that best satisfies the heuristic for way-point navigation (see Section 5.2.1). These errors are guaranteed to never occur if the time horizon is set to ∞ and the robot starts in a valid initial condition. Limited vehicle dynamics do not affect this guarantee because simply continuing at the current safe velocity is always dynamically feasible.

- **VOS numerical errors** The circumstantial geometry of the boundary of the VOS of a certain obstacle may occasionally be problematic for the algorithm to process. Two boundary points may be too close to cleanly distinguish numerically, or multiple candidate boundary segments may overlap in a very near parallel manner that makes it problematic to find the overall outer boundary. In these cases, the VOS is not properly computed, and collisions may result. Empirically, the VOS calculation fails less than 0.5% of the time, and this rate can perhaps be reduced with further refinement in the coding of the algorithm. Errors of this type should be disregarded when evaluating the effectiveness of the collision avoidance algorithm under various conditions.

These errors do not always result in actual collisions, since the obstacles may end up moving along relatively benign trajectory among all its dynamically feasible choices. Therefore, the error rates and collision counts are recorded separately.

Measures of Effective Path Planning The direct distance between successive way-points is $6\sqrt{2} \approx 8.5$ meters, so by traveling at the maximum speed and ignoring all obstacles, the host robot would need ≈ 3.4 seconds to travel from way-point to way-point. Accounting for the collision avoidance constraints in the form of the velocity obstacle sets forces the host robot to divert from this direct path and interferes with the goal-seeking process. The average time it actually takes for the robot to reach the way-point in each simulation is recorded as a measure how effectively of the path-planner can navigate the way-points while operating within the various imposed safety constraints.

Empirically, when a way-point is not reached immediately, the host robot typically needs to circle around the environment before it can make another pass to the same way-point. It is also fairly common for multiple obstacles to linger in the vicinity of a way-point such that the robot must wait a significant duration before it is possible to reach safely approach the goal. These situations give rise to significant outliers in the time-to-goal data. The standard deviation is be calculated, but the distribution is not normal, so its characteristics are not summarily represented by the average and standard deviation alone. Therefore, a distribution of times is also recorded for each simulation; the fraction of paths that are completed within various multiples of the direct path time (≈ 3.4s) is recorded.

Infinite Horizon Data In the figures in which these measurements are plotted (Figures 5-14-5-17 and 5-21-5-24), the infinite horizon simulation is in the cases without walls, and the data for $\tau = \infty$ is plotted at some finite value (24 or 30) at the far right of the x axis.

5.4.2 Results: No walls, No dynamic constraints

In this batch of tests, like the case in the baseline implementation of Section 5.3, the environment is not enclosed by walls, and the host vehicle can turn instantaneously to any new velocity. Figures 5-11 to 5-13 show the changes in the VOS constraints as the time horizon is increased. On short horizons, only the obstacles in the immediate vicinity of the host robot affect the safety of feasible velocities. The planner essentially acts as a reactive planner and does not make use of the VOS formulation to account for the unpredictable future behaviors of the obstacles. Nonetheless, the VOSs still do accurately represent which velocities are necessary to steer away from immediate collisions, and as long as the host robot is agile enough to veer away, collisions can generally be avoided. Short horizon planners could be vulnerable to becoming gradually surrounded, by the obstacles, but such a pathological situation did not arise during the simulation. As the horizon is extended, the planner becomes more sensitive to possible future locations of the obstacles, and would be susceptible to encountering situations that require a last minute dodging behavior. This increased awareness of future obstacle trajectories imposes more restrictive limits on what are considered safe velocities; any velocity deemed safe on a longer horizon is also deemed safe on a shorter one. Therefore, in general, if the heuristic for selecting velocities that get to the way-point quickly is effective, as the time horizon is increased, the average time to goal should rise. After some point, further extending the time horizon has negligible effects on the shape of the VOSs, so the performance measures should settle as $\tau \to \infty$.

Figures 5-14 to 5-17 confirm these trends. With an unlimited turn rate, the host robot does fine using even just a reactive planner, so there are no errors or collisions on any time horizon (the VO errors are numerical issues, as explained in Section 5.4.1). As the time horizon is increased, the average time to goal climbs. There is a minor peak in Figure 5-16 at $\tau = 6$s, and this may be caused by transients in switching between two qualitatively different behaviors: reacting to imminent collisions versus planning more roundabout paths using an essentially infinite horizon. For horizons

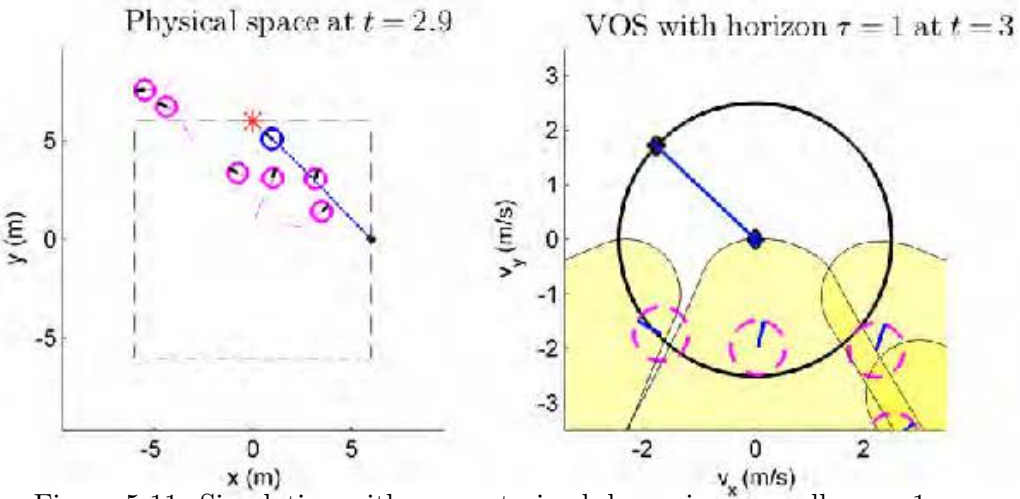

Figure 5-11: Simulation with unconstrained dynamics, no walls, $\tau = 1$s.

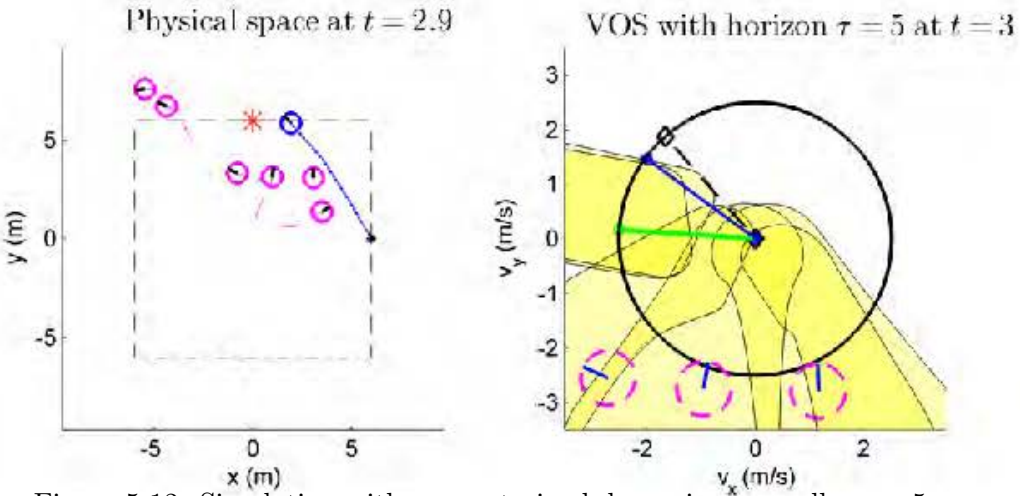

Figure 5-12: Simulation with unconstrained dynamics, no walls, $\tau = 5$s.

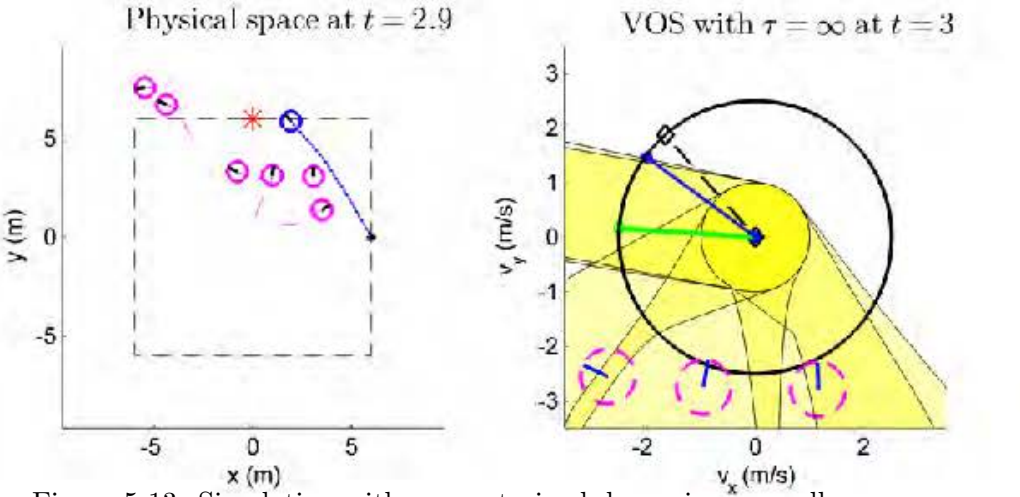

Figure 5-13: Simulation with unconstrained dynamics, no walls, $\tau = \infty$.

greater longer 10 seconds, the behavior is essentially identical to the infinite horizon behavior. With an unbounded turn rate, if the host robot misses the way-point, it can immediately turn around without making a large loop. Therefore, as seen in Figure 5-17, there are not very many extremely long paths.

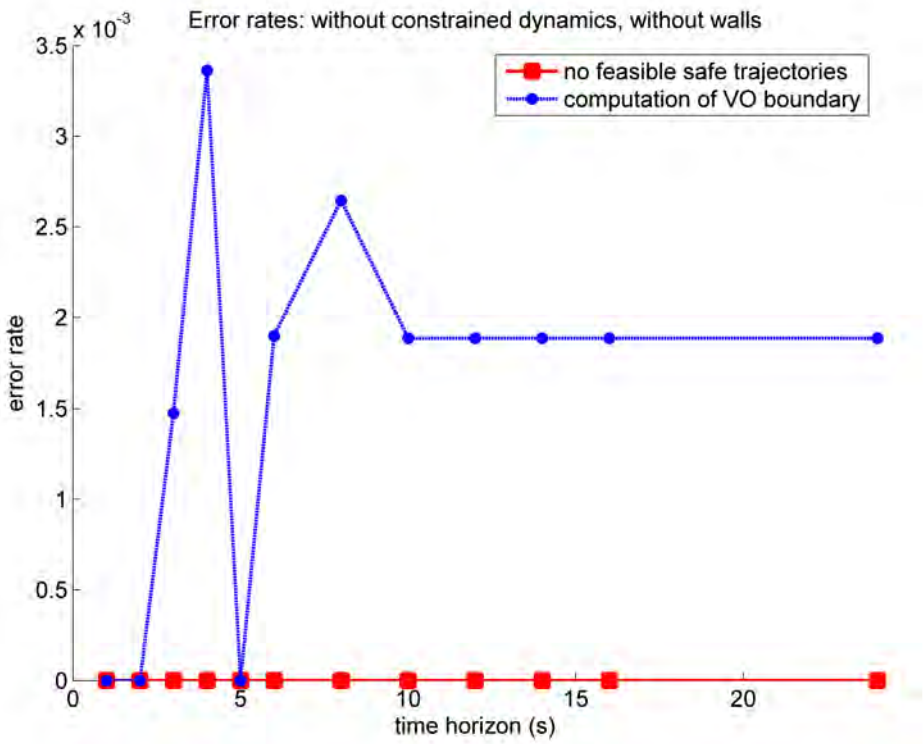

Figure 5-14: Errors: no constraints, no walls

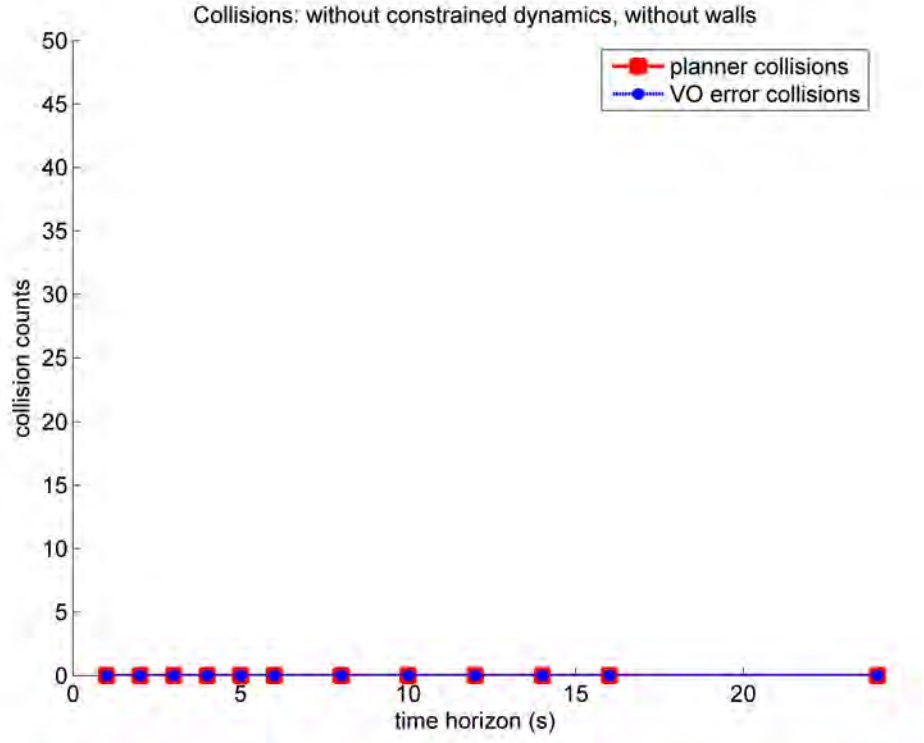

Figure 5-15: Collisions: no constraints, no walls

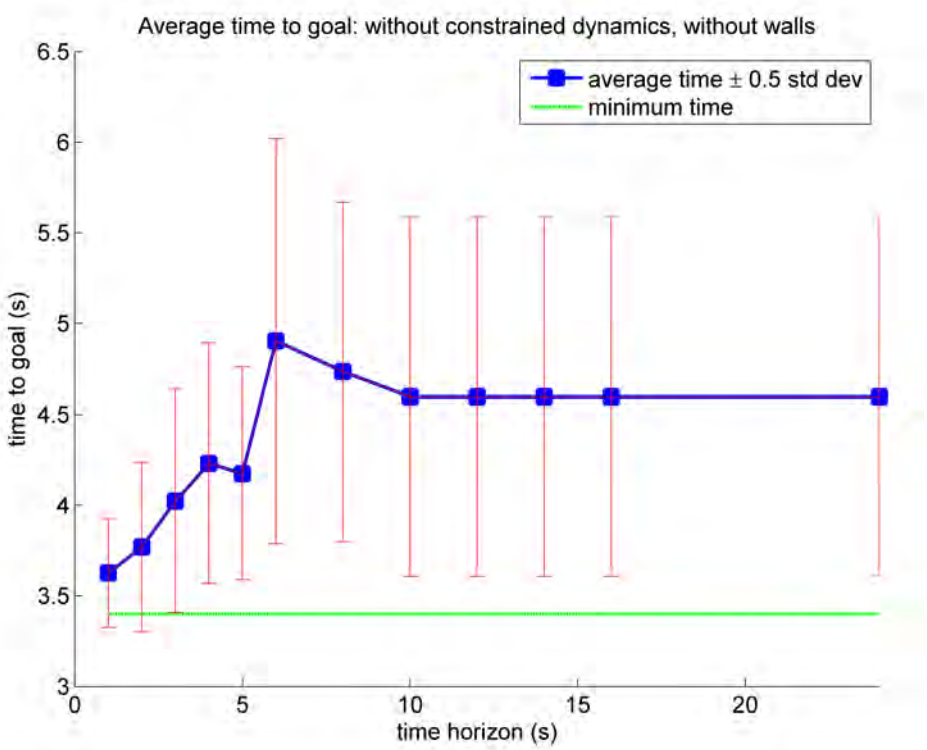

Figure 5-16: Average time to goal: no constraints, no walls

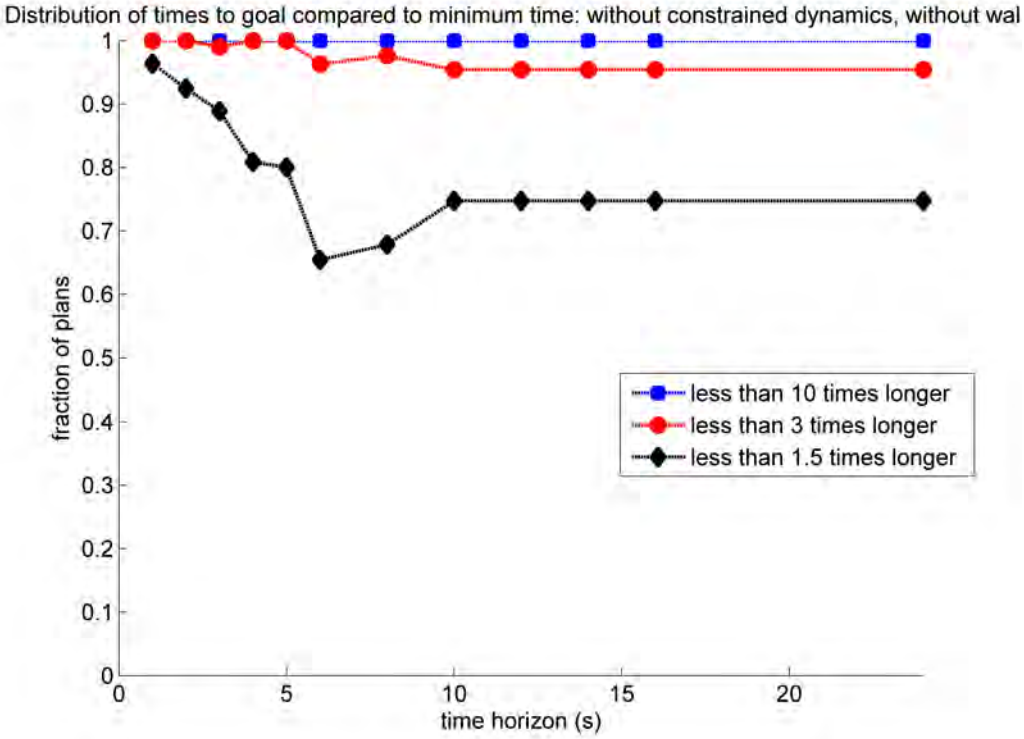

Figure 5-17: Distribution of time to goal: no constraints, no walls

5.4.3 Results: No walls, Dynamic constraints

In this batch of tests, a turn-rate constraint is introduced for the host robot such that, at each planner update, it may only choose velocities whose angles differ by at most $\Delta\theta = \pi/3$ from the previous velocity (see section 5.2.2). In Figures 5-18 to 5-20, these constraints are represented by the red bounds (which are bisected by the previous velocity, the dashed black vector): only velocities from within the wedge may be chosen as the new velocity.

In these scenarios, the robot is less agile, so reactive motion planning is naturally less effective. The short-term planner may attempt a direct path towards the way-point, and along the way, it then discovers a threatening obstacle but not be able to swerve avoid it in time. Such situations indeed led to multiple collisions for the 1s horizon planner (Figure 5-22), and threatened other collisions that were avoided only by chance (Figure 5-21). This issue is provably averted only through using the infinite horizon planner, though in practice, the likelihood of the problem drops off sharply as longer finite horizons are used. The limited dynamics of the host vehicle do not affect the infinite horizon safety guarantee.

Just as in the unconstrained case, if the velocity selection heuristic is effective, shorter planning horizons would generally imply shorter times to goal, since there are fewer restricted velocities. However, with limited turning dynamics on the host vehicle, the times are noisier and this trend is less pronounced. Here, the heuristic will often fail; the vehicle cannot greedily seek the goal and successfully make quick adjustments along the way, if the trajectory is deflected by an obstacle, the host vehicle needs to slowly loop back around before its again aligned with the way-point. Thus, executing a plan that looks deceptively good on a short horizon can be more costly, and these competing effects give rise to widely scattered data in Figure 5-23.

All together, the times are much longer in this scenario compared to the previous one. This comes from the fact that, without considering more general safe states and multi-step paths (see Sections 4.3 and 4.4), the planner will have a lot of trouble approaching the goal with vehicles on the other side of the way-point, due to the

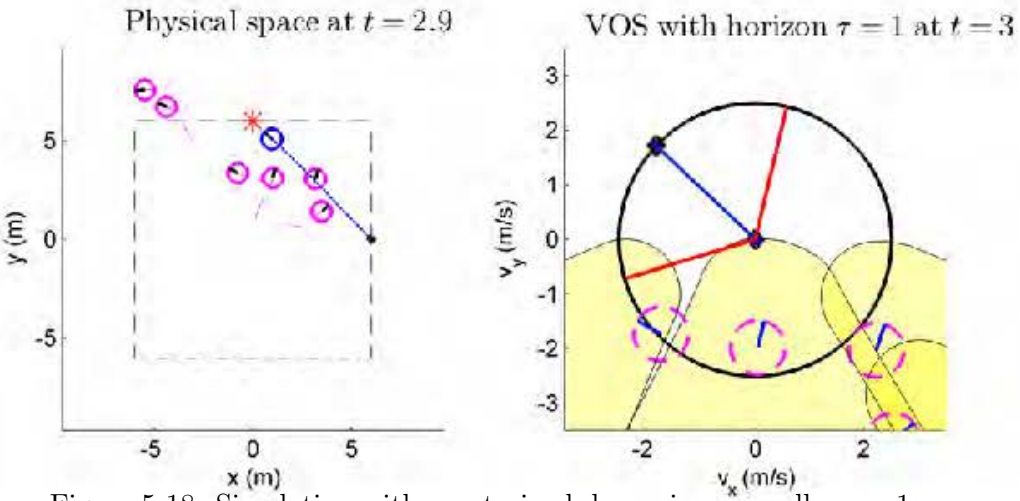

Figure 5-18: Simulation with constrained dynamics, no walls, $\tau = 1$s.

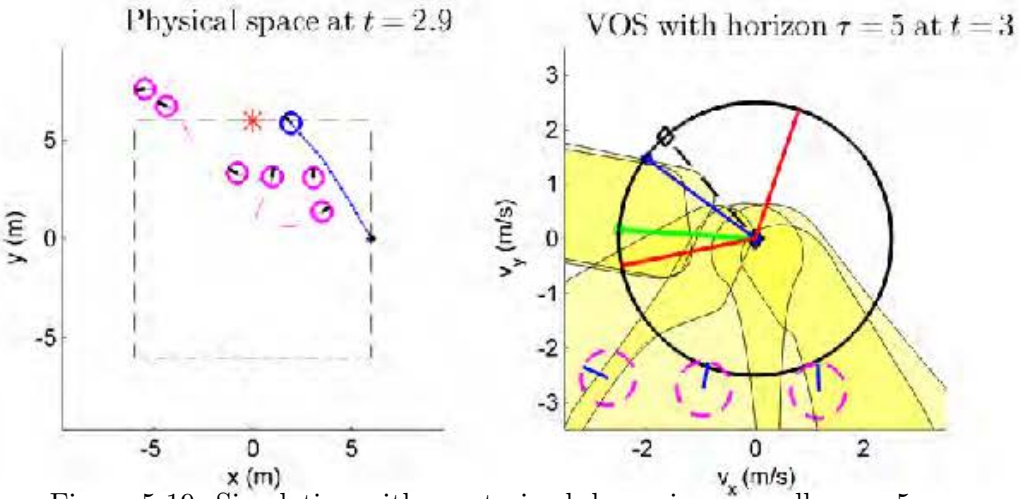

Figure 5-19: Simulation with constrained dynamics, no walls, $\tau = 5$s.

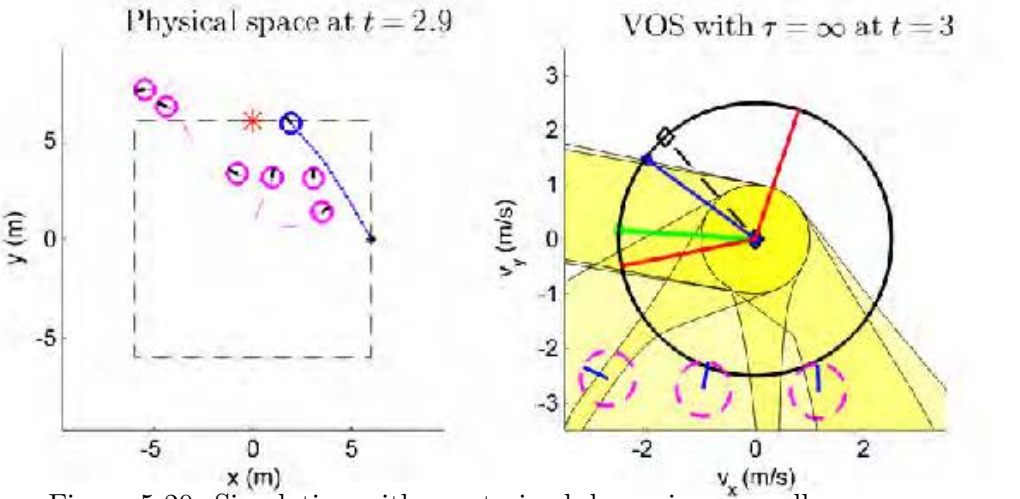

Figure 5-20: Simulation with constrained dynamics, no walls, $\tau = \infty$.

fact that the linear trajectory must be continued safely (see Figures 5-25 and 5-26). In the previous scenario, if the robot misses the way-point for this reason, usually it can safely immediately double back to reach the goal, but it is unable to do so with a limited turn-rate. The inability to take such a corrective action causes the much longer paths (Figures 5-23 and 5-24). This effect would be significantly reduced if the planner were capable of considering partial trajectories towards the goal that later along safe infinite trajectories in a different direction; even the dynamically limited planner would be able to have a higher success rate on the first passes towards the way-point.

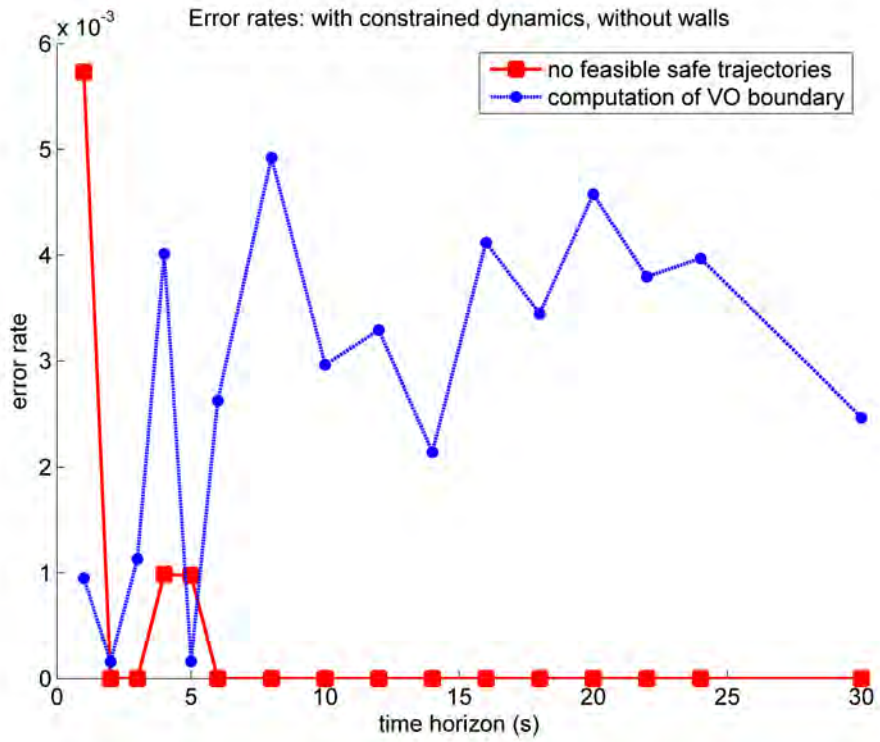

Figure 5-21: Errors: with constraints, no walls

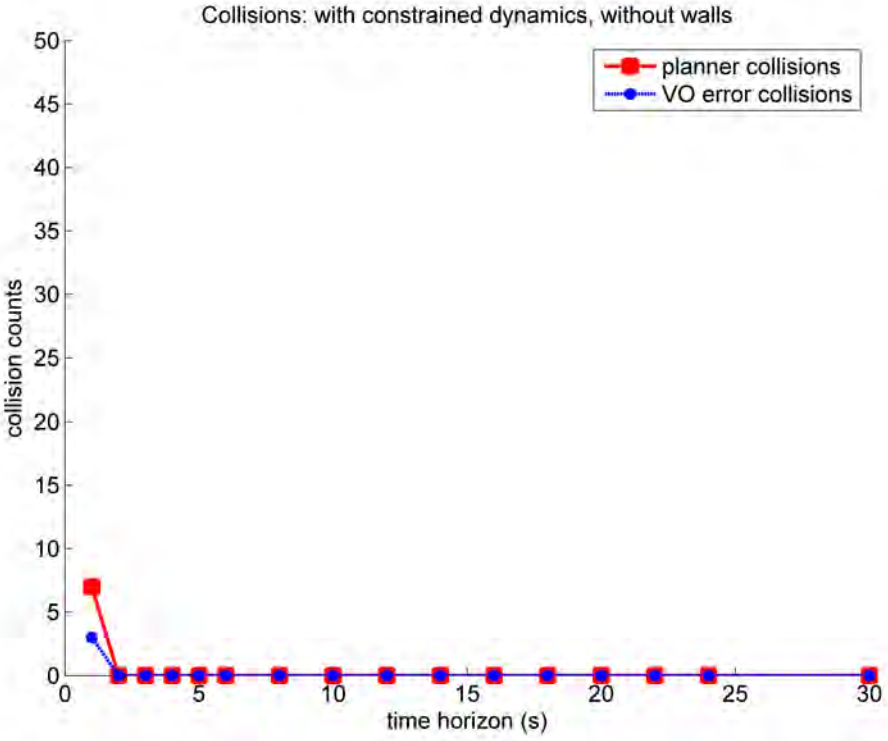

Figure 5-22: Collisions: with constraints, no walls

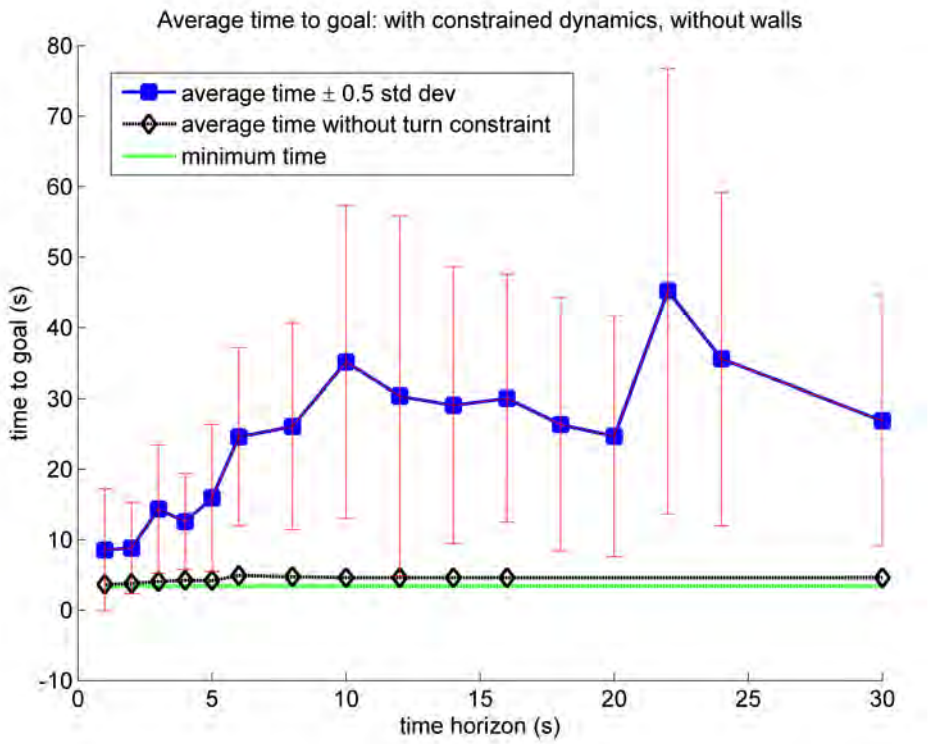

Figure 5-23: Average time to goal: with constraints, no walls

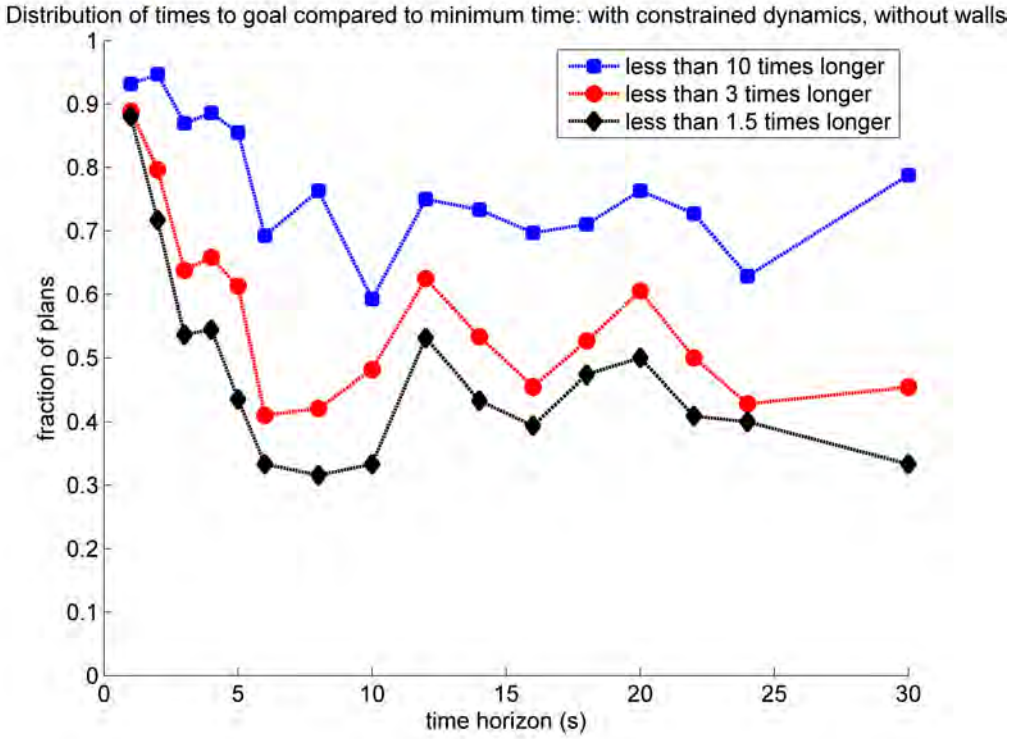

Figure 5-24: Distribution of time to goal with constraints, no walls

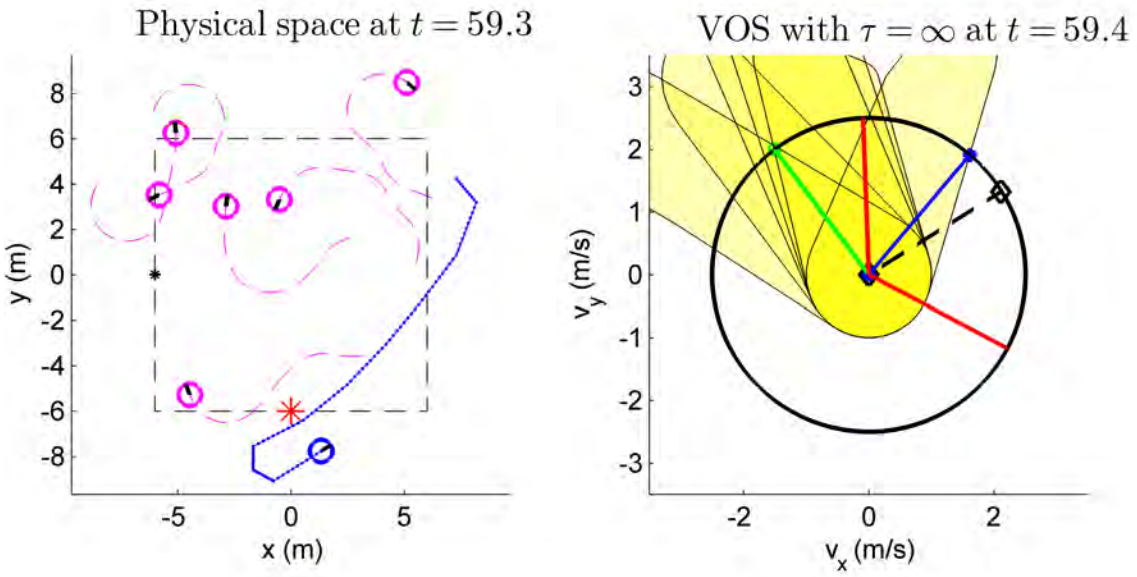

Figure 5-25: Were the planner able to consider more general trajectories, it could find a safe path that goes directly to the goal with the intention of later merging onto a safe infinite trajectory pointed towards the right. The inability to consider such a trajectory hampers both planners, with or without turn-rate constraints.

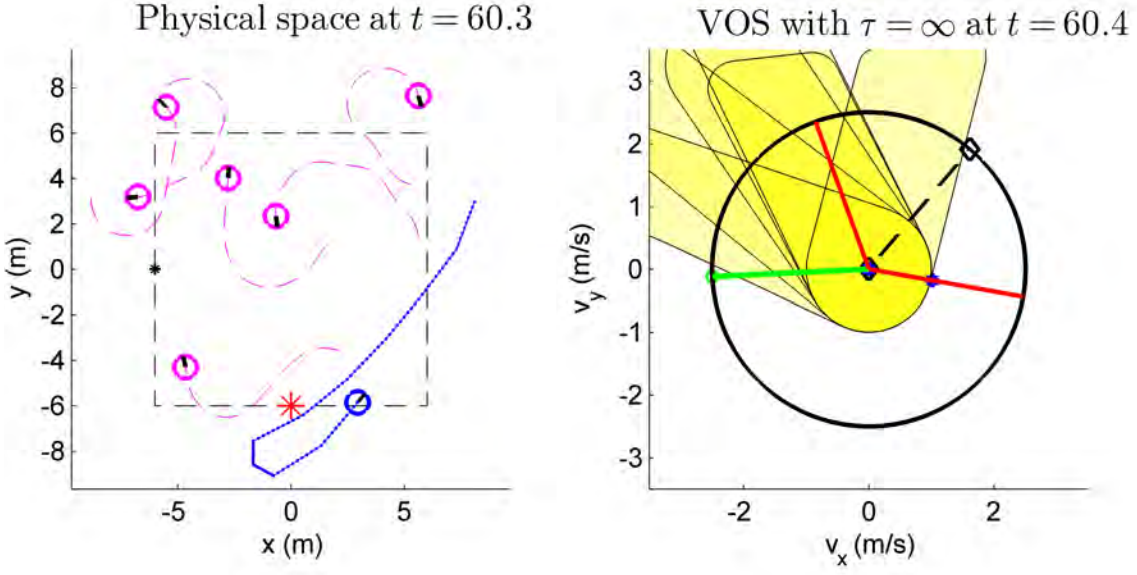

Figure 5-26: However, without turn rate constraints, the host vehicle would have been able to turn back immediately to take the green trajectory. In this case, it cannot do so, and it is often left "pacing" back and forth.

5.4.4 Results: Walls, No dynamic constraints

In this batch of tests, the environment is enclosed by walls (at the limit of the viewing window of the left sides of Figures 5-27 to 5-29), and the host vehicle can turn instantaneously to any new velocity. The velocities that would result in a collision with the wall at time $t = \tau$ are represented by the red boxes; velocities outside of the box are not τ-safe and are not chosen by the planner. Note that, with multiple obstacles, the required conditions to be able to guarantee collision avoidance are quite complex. In a pathological scenario, depending on the size of the room and the relative dynamics of the vehicles, it may be possible for the obstacles to collectively herd the host vehicle into a corner such that collision is inevitable. This situation does not arise in simulation.

On an infinite time horizon, any non-zero velocity would lead to a collision with a wall, so the infinite-horizon planner is not considered here. For short time horizons, unless the host vehicle is very close to a wall, the reactive planner is essentially unaffected. As the time horizon is increased, avoiding the walls becomes an increasingly restrictive requirement that essentially limits the host vehicle to lower and lower speeds. This effect can be seen in the left halves of Figures 5-32 and 5-33. Meanwhile, increasing the time horizon also drives each velocity obstacle set towards cover the entire unit circle (as explained in Section 3.3.2). Past roughly $\tau = 10$, all the velocities within the red box imposed by the walls are usually covered by the VOSs such that, more often than not, the planner fails to find any single-velocity trajectories that are safe for duration τ (Figure 5-30).

As explained in Section 5.2.1, when this occurs, the planner is allowed to select an arbitrary velocity, since there is no way to properly satisfy the safety constraint. By $\tau = 16$, the error rate reaches 100% and the host robot is driving blindly; it heads straight for the way-point and often comes in contact with the obstacles (Figure 5-31. This planner's performance may be treated as the "control" case where nothing is done.

These results show that, when operating in an enclosed environment, it does not

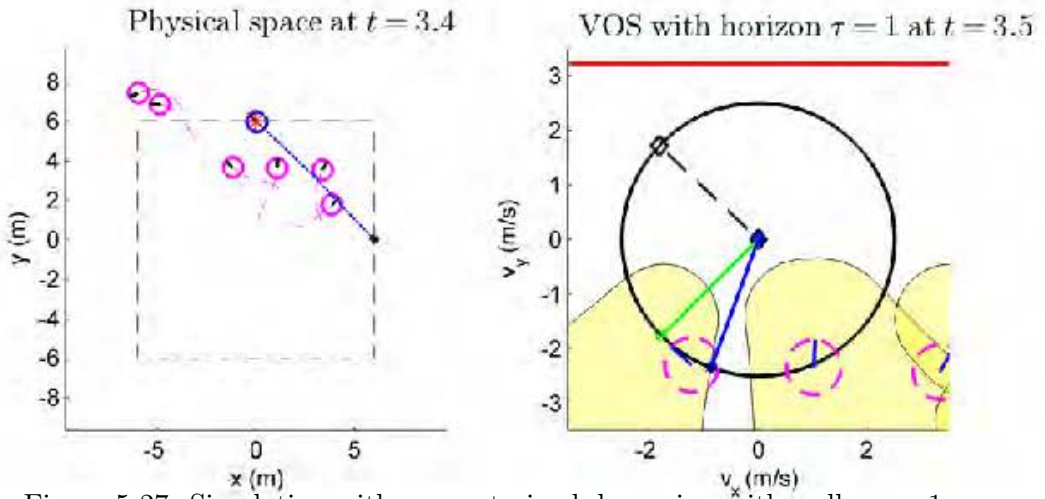

Figure 5-27: Simulation with unconstrained dynamics, with walls, $\tau = 1$s.

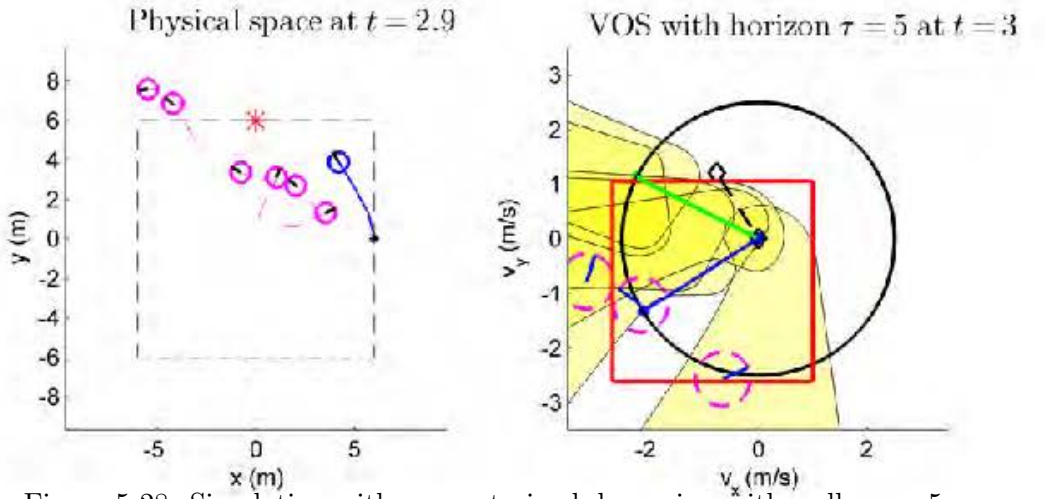

Figure 5-28: Simulation with unconstrained dynamics, with walls, $\tau = 5$s.

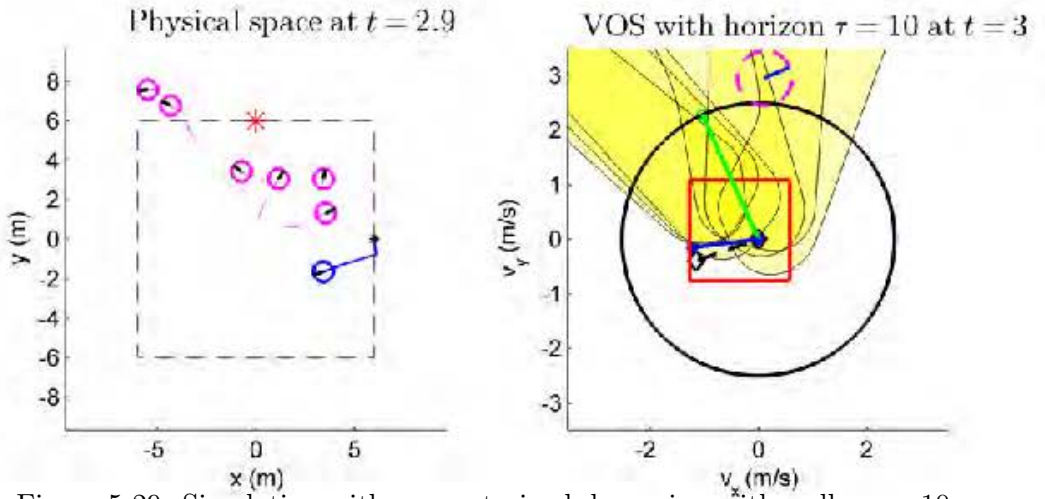

Figure 5-29: Simulation with unconstrained dynamics, with walls, $\tau = 10$s.

make sense to use long time horizons, since no single-velocity trajectory stays safe very long. With agile dynamics, short term planners can still be effectively implemented to reactively dodge obstacles while avoiding swerving into the wall.

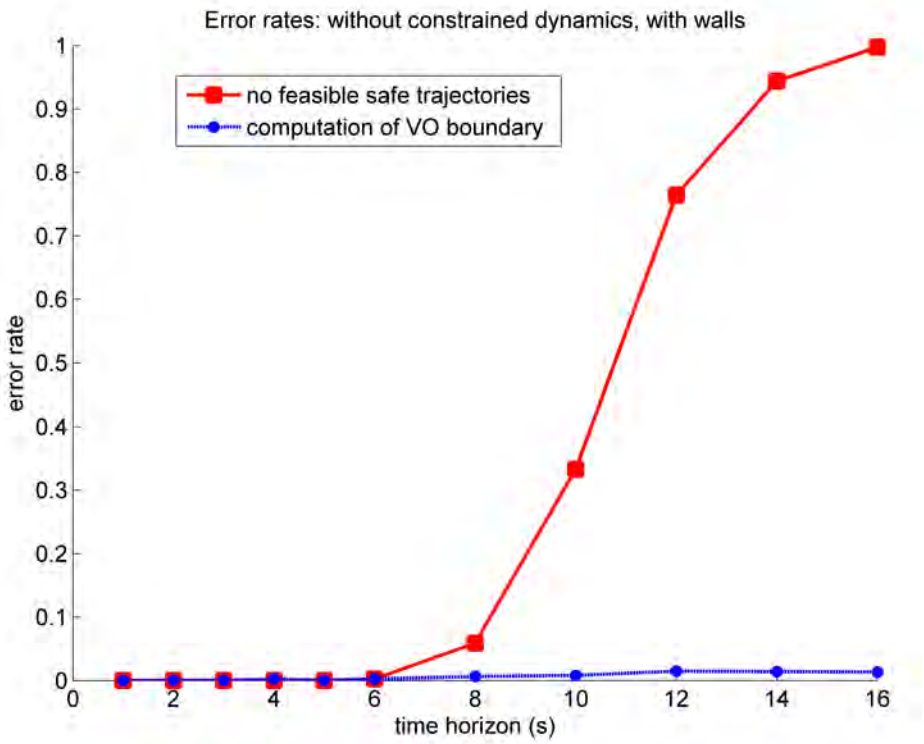

Figure 5-30: Errors: no constraints, with walls

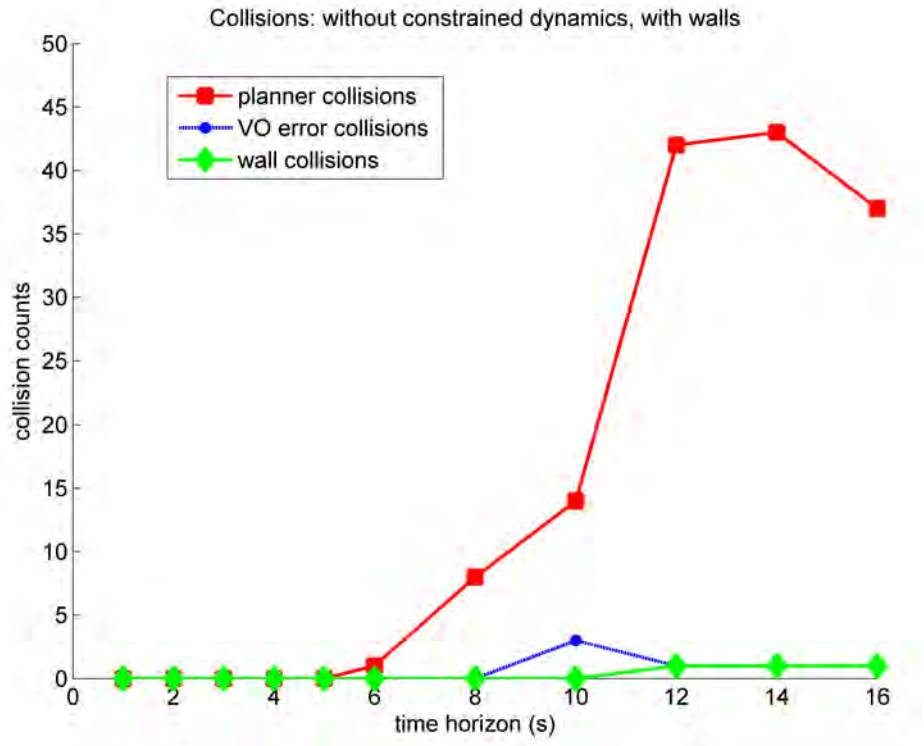

Figure 5-31: Collisions: no constraints, with walls

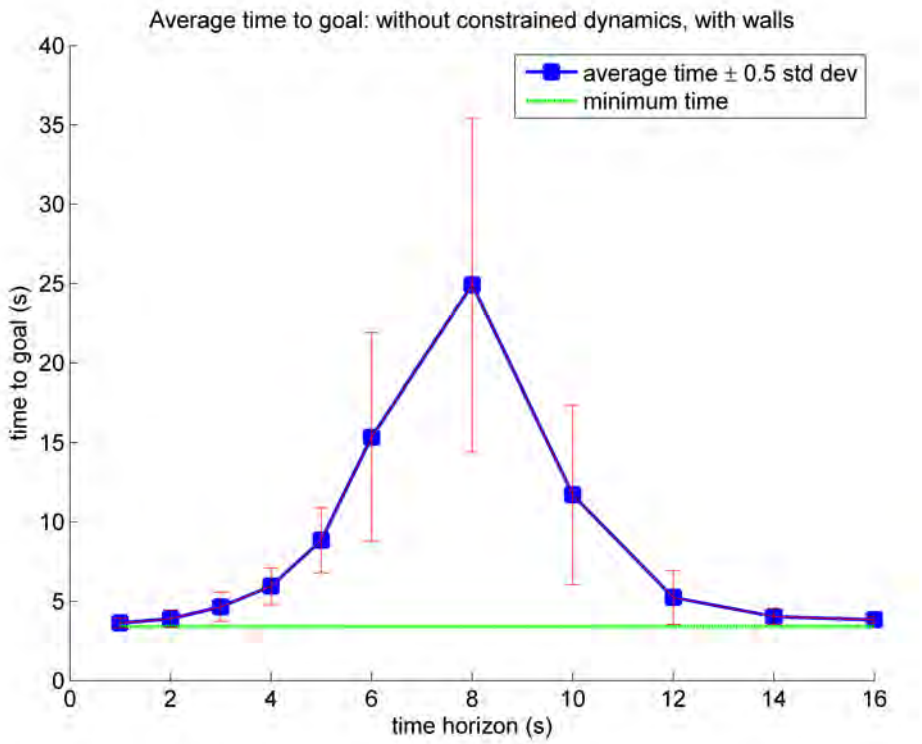

Figure 5-32: Average time to goal: no constraints, with walls

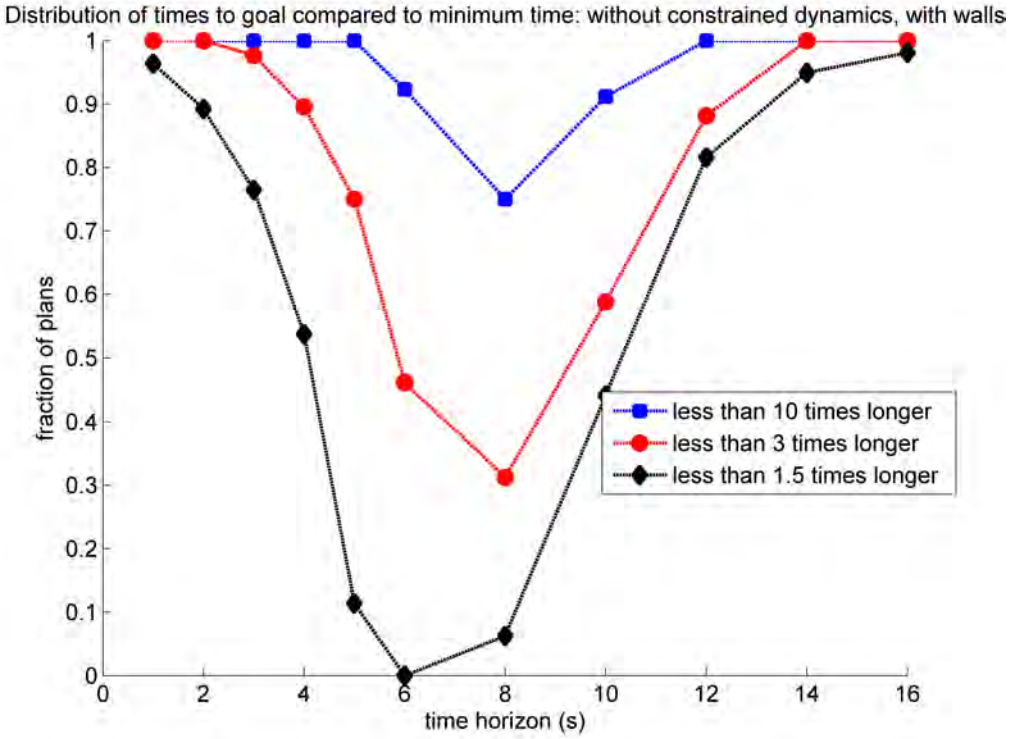

Figure 5-33: Distribution of time to goal: no constraints, with walls

5.4.5 Results: Walls, Dynamic constraints

In this batch of tests, the environment is enclosed by walls, and the host vehicle has a limited turn rate. In Figures 5-34 to 5-36, valid velocities must lie inside both the wedge defined by the limited dynamics and the box defined by the walls.

As one would expect, this batch of simulation results is a combination of the previous two cases. Once the time horizon gets too long, there are no safe trajectories, and the vehicle reverts to driving without collision avoidance. This transition time horizon occurs a little earlier than it did without dynamic constraints, since the added angular restriction increases the likelihood that no safe choices are available (as is the case in Figure 5-36). Before the transition occurs, the turn-rate restricted planner is not nearly as effective as the unconstrained one, both in terms of avoiding collisions (Figure 5-37) and in terms of planning efficient paths to the way-points (Figure 5-39). At $\tau = 8$s, the restricted planner is on average faster, but this is due to the switch in behavior occurring earlier at $\tau = 6$s.

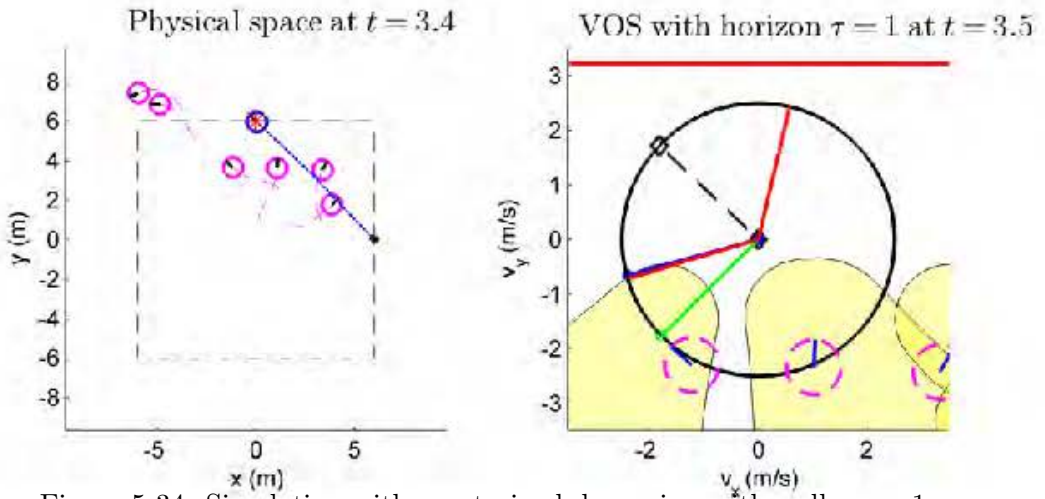

Figure 5-34: Simulation with constrained dynamics, with walls, $\tau = 1$s.

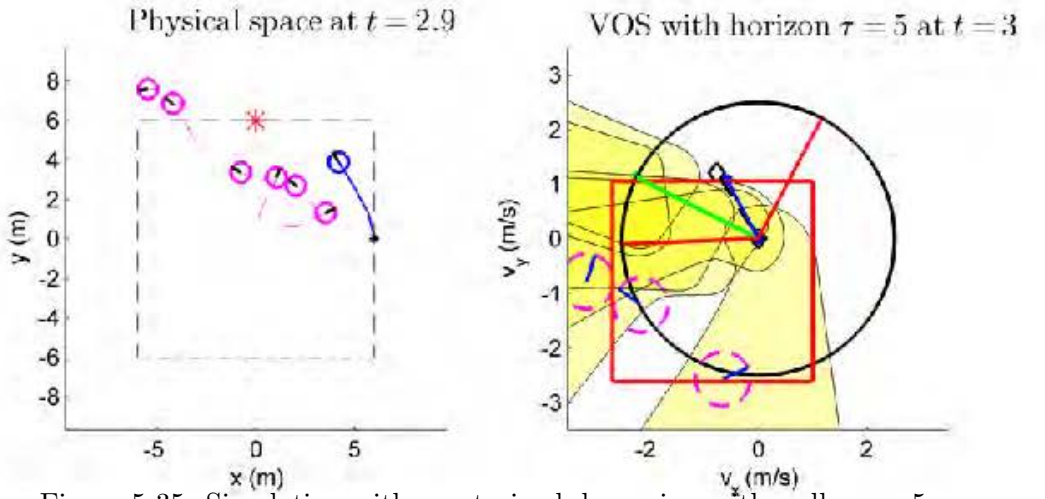

Figure 5-35: Simulation with constrained dynamics, with walls, $\tau = 5$s.

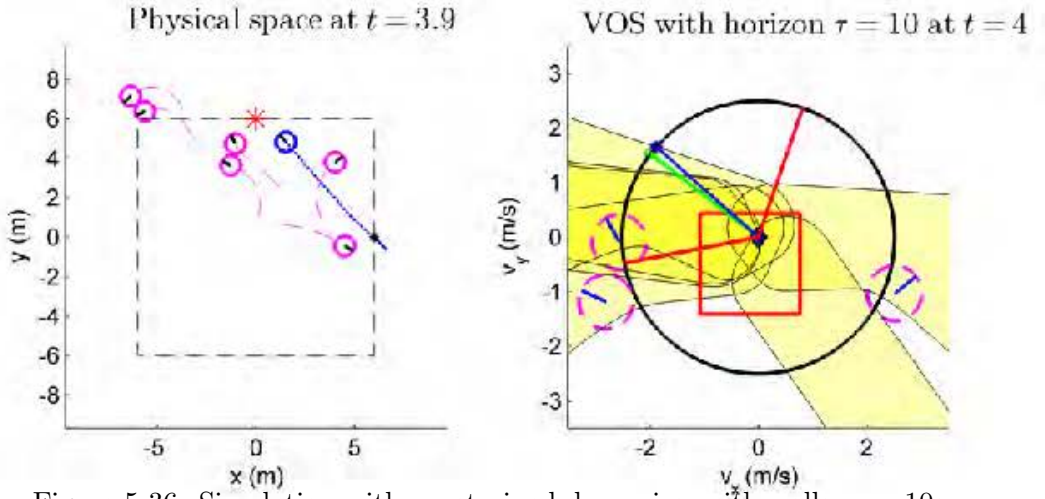

Figure 5-36: Simulation with constrained dynamics, with walls, $\tau = 10$s.

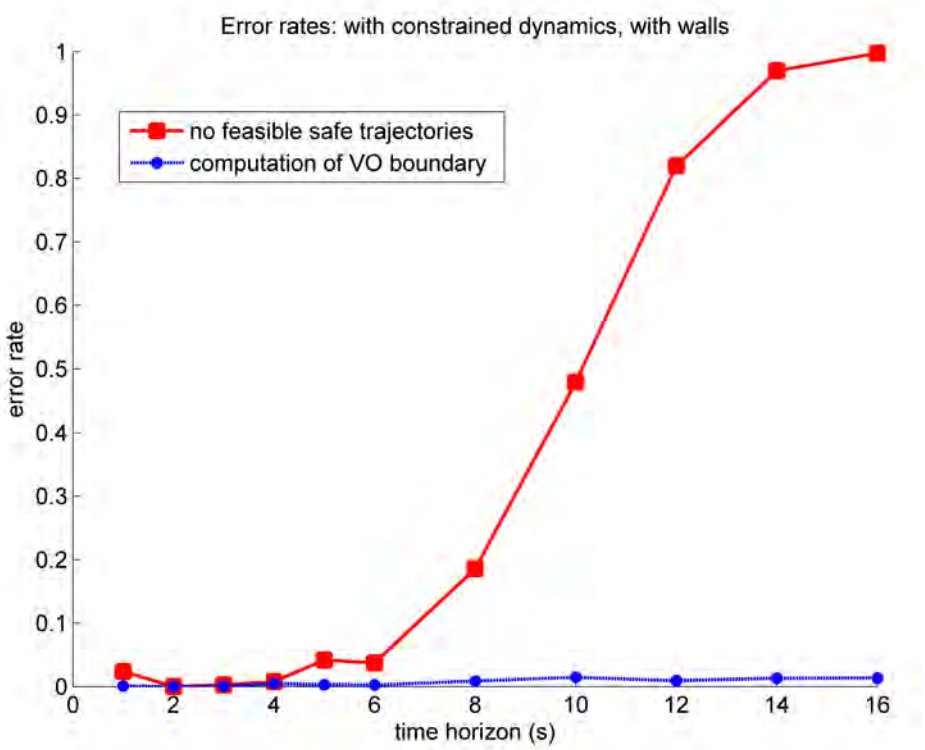

Figure 5-37: Errors: with constraints, with walls

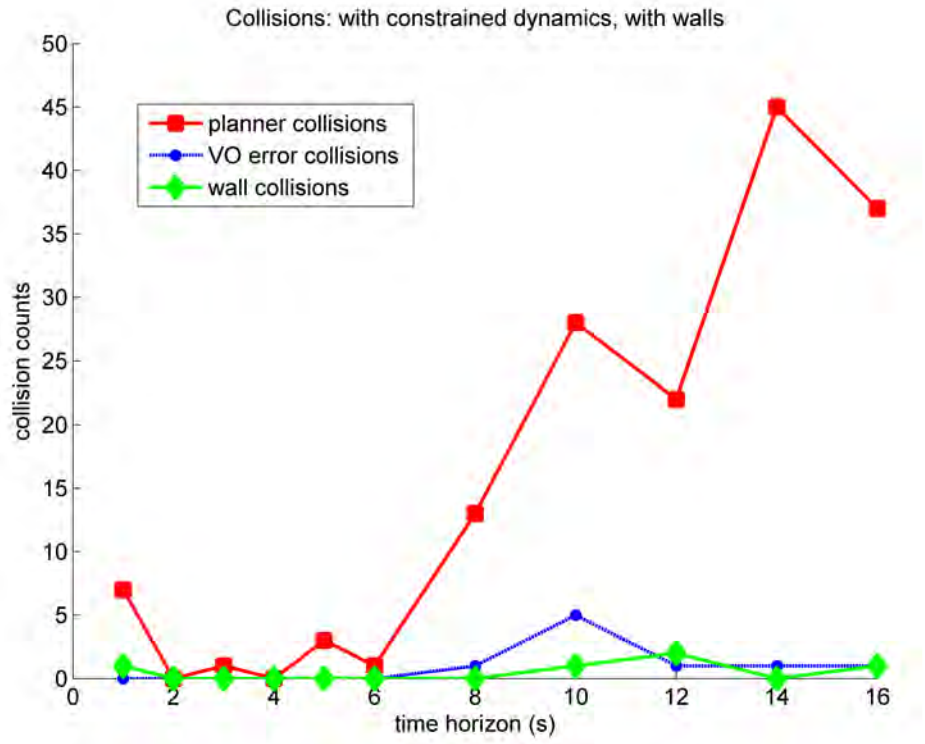

Figure 5-38: Collisions: with constraints, with walls

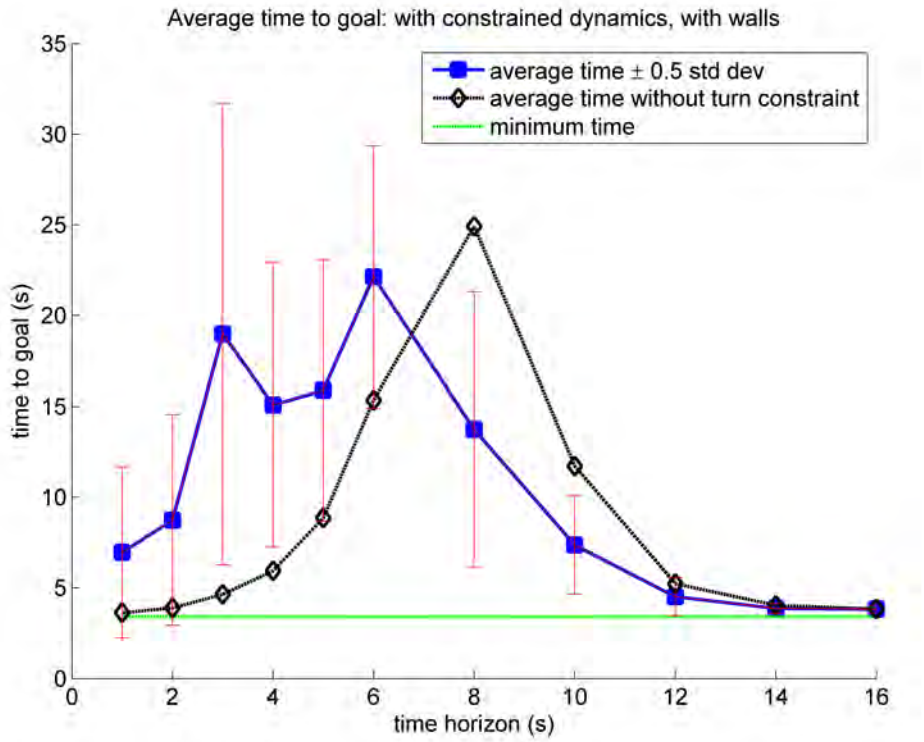

Figure 5-39: Average time to goal: with constraints, with walls

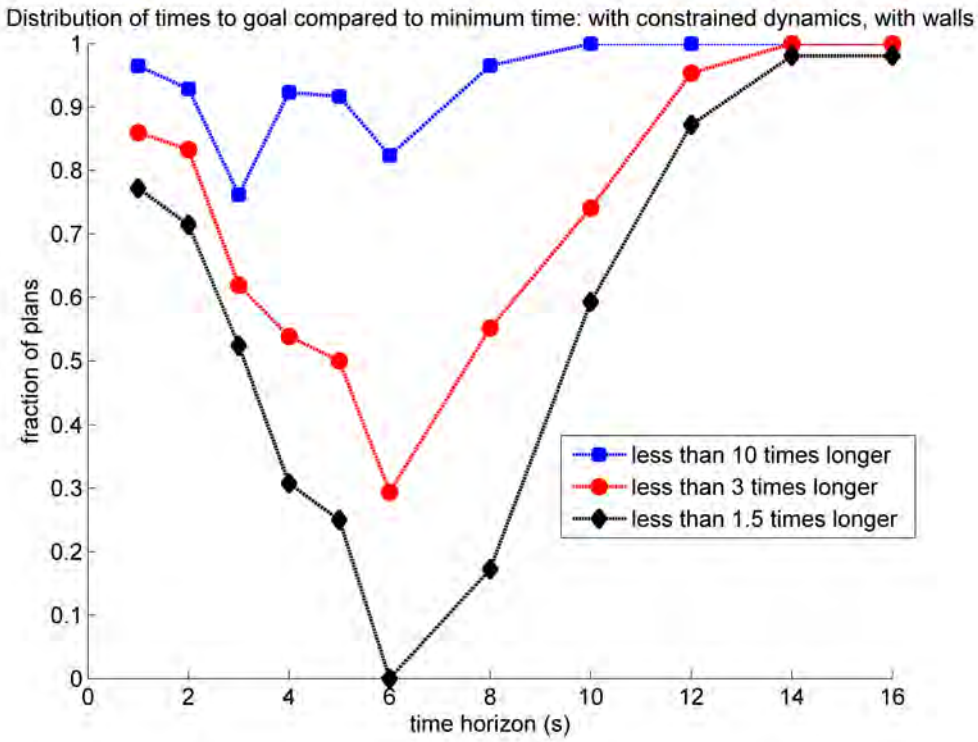

Figure 5-40: Distribution of time to goal: with constraints, with walls

Chapter 6

Conclusions

The objective of this thesis is to develop a framework for guaranteeing infinite horizon collision avoidance of unpredictable, dynamically constrained moving obstacles. This goal is achieved by converting the reachable sets of the unpredictable obstacles into velocity space constraints, whose union over a window of time defines the velocity obstacle set. As the upper limit of the time window is extended to infinity, this velocity obstacle set can be used to find safe, invariant single-velocity escape trajectories that can be used in a receding horizon setting to provide guaranteed safety.

6.1 Thesis Summary

The objective of this thesis is introduced and motivated in Chapter 1. In better structured environments, invariant safe states are often used as a cornerstone in proving safety in receding horizon settings, but such states have been difficult to define in the presence of external agents with unpredictable behavior. Meanwhile, velocity obstacles are widely used to concisely represent conditions for guaranteed avoidance of predictable obstacles. However, these approaches are heavily dependent on having accurate information about the obstacles' future locations, which is typically an invalid assumption. The algorithm developed in this thesis does not require such an assumption, and uses the velocity obstacle concept to define invariant safety in velocity space, thereby successfully guaranteeing infinite horizon safety in this challenging

setting.

In Chapter 2, parametric equations are found to describe the boundaries of *simplified collision regions* that enclose all potentially dangerous locations with respect to each obstacle as a function of time. In Chapter 3, these collision regions are mapped into velocity space, and the union of the resulting *simplified velocity regions* over a time interval defines the *velocity obstacle set*. Velocities that are not within the velocity obstacle set can be safely propagated along single-velocity trajectories, avoiding all potential collision locations for the duration of the interval. It is shown that the velocity obstacle set can be legitimately defined for an open time interval that extends to $+\infty$.

Thus, as laid out in Chapter 4, the infinite horizon velocity obstacle set can be used to define invariant safe states with respect to the unpredictable dynamic obstacles in the form of single-velocity infinite trajectories. These safe velocities can be immediately used as the raw building blocks of a rudimentary iterative planner with infinite horizon guarantees, or they can be further refined to define more general safe states that would serve as the endpoints of arbitrary trajectories which could then be used in more advanced iterative planners that also preserve guaranteed infinite horizon collision avoidance.

Finally, the performance of iterative planners using velocity obstacle sets of various time horizons are presented in Chapter 5. Only the infinite horizon planner is guaranteed safe, but simulation results show that, if the host robot has sufficient mobility, short horizon velocity obstacles can be used in planners that essentially use reactive collision avoidance. Such planners also tend to be more successful in finding direct paths to the goal. Furthermore, they can be used in confined environments in which long horizon planners are invalid. However, the short horizon planners exhibit non-zero error rates, and are notably less effective and more risky when the host robot's dynamics are limited. The simulation results also suggest that performance in all cases can be drastically improved with generalized safe states that would allow for more intelligent trajectory construction.

6.2 Future work

The concept of Velocity Obstacle Sets and the resulting path planning algorithm can be expanded and improved in several directions with additional research. All the derivations presented in this thesis apply to obstacles with assumed unicycle dynamics, i.e., the obstacles move with constant forward speed and have a bounded turn-rate. Additional work can be done to extend the result to obstacles defined by a different set of dynamics. The reachable set for objects with a maximum speed instead of a single constant speed should look quite similar to the reachable sets discussed here, so an extension to cover this more general case may not be too difficult to derive. It may also be worthwhile to consider the problem more carefully from the parallel perspective of inevitable collision states (Section 2.3.1).

In computing the boundary of the velocity obstacle set, the process of combining various candidate points together into a continuous curve is not yet completely robust. It relies heavily on basic functions that check for intersections between arbitrary line segments, but this procedure can break down when the input line segments extremely close and are nearly end to end, parallel, or just barely overlapping. In the current implementation, these issues crucially interfere with the computation of the boundary in about 0.3% of the attempted computations. A careful review of this process or an attempt to find an alternative means of processing the candidate boundary points may reduce or eliminate this numerical issue.

Furthermore, in implementation, this is the most computationally intensive step. It can require up to half a second (though on average much less) to process a boundary of a few hundred points on a desktop computer, whereas all the other steps combined require a fraction of that time. Conceptually, velocity obstacle sets are perfectly suited for use in real-time on-board planning algorithms, so for this purpose as well, improving the above described process would be a worthy pursuit.

Finally, Chapter 4 lays out the process for finding potential generalized safe states using time-shifted velocity obstacle sets. These would allow iterative planners to properly handle the unavoidable host robot non-linearities of turning, and to create

far more effective plans using finite trajectories terminating in safe states that have infinite horizon safe solutions, as discussed in Chapter 5. The implementation of an algorithm for computing the boundary of the time-shifted velocity obstacle sets would therefore be a useful future project.

Bibliography

[1] Y. Kuwata, G. A. Fiore, J. Teo, E. Frazzoli, and J. P. How, "Motion planning for urban driving using RRT," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 22-26 Sept. 2008, pp. 1681–1686. [Online]. Available: 10.1109/IROS.2008.4651075

[2] Y. Kuwata, S. Karaman, J. Teo, E. Frazzoli, J. How, and G. Fiore, "Real-time motion planning with applications to autonomous urban driving," *Control Systems Technology, IEEE Transactions on*, vol. 17, no. 5, pp. 1105–1118, 2009.

[3] A. Chakravarthy and D. Ghose, "Obstacle avoidance in a dynamic environment: A collision cone approach," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 28, no. 5, pp. 562–574, 1998.

[4] E. Cockayne and G. Hall, "Plane motion of a particle subject to curvature constraints," *SIAM Journal on Control*, vol. 13, p. 197, 1975.

[5] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *The International Journal of Robotics Research*, vol. 17, no. 7, p. 760, 1998.

[6] J. Reif and M. Sharir, "Motion planning in the presence of moving obstacles," in *Foundations of Computer Science, 1985., 26th Annual Symposium on.* IEEE, 1985, pp. 144–154.

[7] R. Murray and S. Sastry, "Nonholonomic motion planning: Steering using sinusoids," *Automatic Control, IEEE Transactions on*, vol. 38, no. 5, pp. 700–716, 1993.

[8] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *Robotics & Automation Magazine, IEEE*, vol. 4, no. 1, pp. 23–33, 1997.

[9] E. Frazzoli, M. Dahleh, and E. Feron, "Real-time motion planning for agile autonomous vehicles," *Journal of Guidance Control and Dynamics*, vol. 25, no. 1, pp. 116–129, 2002.

[10] A. Richards and J. How, "Aircraft trajectory planning with collision avoidance using mixed integer linear programming," in *American Control Conference, 2002. Proceedings of the 2002*, vol. 3. IEEE, 2002, pp. 1936–1941.

[11] J. Latombe, "Robot motion planning, chapter," 1996.

[12] E. Prassler, J. Scholz, and P. Fiorini, "A robotics wheelchair for crowded public environment," *Robotics & Automation Magazine, IEEE*, vol. 8, no. 1, pp. 38–45, 2001.

[13] P. Fiorini and Z. Shiller, "Time optimal trajectory planning in dynamic environments," in *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, vol. 2. IEEE, 1996, pp. 1553–1558.

[14] R. Simmons, "The curvature-velocity method for local obstacle avoidance," in *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, vol. 4. Ieee, 1996, pp. 3375–3382.

[15] F. Large, C. Laugier, and Z. Shiller, "Navigation among moving obstacles using the NLVO: Principles and applications to intelligent vehicles," *Autonomous Robots*, vol. 19, no. 2, pp. 159–171, 2005.

[16] O. Gal, Z. Shiller, and E. Rimon, "Efficient and safe on-line motion planning in dynamic environments," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2009, pp. 88–93.

[17] T. Fraichard and H. Asama, "Inevitable collision states-a step towards safer robots?" *Advanced Robotics*, vol. 18, no. 10, pp. 1001–1024, 2004.

[18] Z. Qu, J. Wang, and C. Plaisted, "A new analytical solution to mobile robot trajectory generation in the presence of moving obstacles," *IEEE Transactions on Robotics*, vol. 20, no. 6, pp. 978–993, 2004.

[19] J. Van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2008, pp. 1928–1935.

[20] M. Tahk, H. Bang, and C. Park, "Multiple aerial vehicle formation using swarm intelligence," in *2003 AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2003.

[21] C. Tomlin, G. Pappas, and S. Sastry, "Conflict resolution for air traffic management: A study in multiagent hybrid systems," *Automatic Control, IEEE Transactions on*, vol. 43, no. 4, pp. 509–521, 1998.

[22] T. Schouwenaars, "Safe trajectory planning of autonomous vehicles," Ph.D. dissertation, Citeseer, 2005.

[23] L. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of Mathematics*, vol. 79, no. 3, pp. 497–516, 1957.

[24] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using the relative velocity paradigm," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 1993, pp. 560–565.

[25] Z. Shiller, F. Large, and S. Sekhavat, "Motion planning in dynamic environments: Obstacles moving along arbitrary trajectories," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 4. IEEE, 2005, pp. 3716–3721.

[26] D. Wilkie, J. van den Berg, and D. Manocha, "Generalized velocity obstacles," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2009, pp. 5573–5578.

[27] B. Damas and J. Santos-Victor, "Avoiding moving obstacles: the forbidden velocity map," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2009, pp. 4393–4398.

[28] D. Vasquez and T. Fraichard, "Motion prediction for moving objects: a statistical approach," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 4. IEEE, 2004, pp. 3931–3936.

[29] M. Bennewitz, W. Burgard, and S. Thrun, "Learning motion patterns of persons for mobile service robots," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2002, pp. 3601–3606.

[30] S. Petti and T. Fraichard, "Safe motion planning in dynamic environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2005, pp. 2210–2215.